Software Generation Method

Technical Field

[0001]

The present invention relates to a software generating method, more particularly to a software generating method using formalization by process algebra in the Lyee methodology.

Background Art

[0002]

Over the last years, various methodologies and techniques have been elaborated and proposed to improve one or many aspects related to the software development life cycle. However, despite the great effort in this research field, the production of clearly understood and modifiable systems is still an ambitious goal and far from reached. This is due, on one hand, to the complexity and the subtlety of software themselves and, on the other hand, to the limitations of the current methodologies. Recently, a new and very promising methodology, called Lyee (trademark), has been proposed. Intended to deal efficiently with a wide range of software problems related to different field, Lyee allows the development of software by simply defining their requirements.

[0003]

Nevertheless, since both the semantics of Lyee generated software together with the process of automatic generation of software from requirements are described using informal

- 1 -

language, difficulties and confusions may arise when trying to understand and study this methodology.

Non-Patent Document 1

J. A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. Theoretical Computer Science, 37(1):77-121, 1985.

Non-Patent Document 2

G. Berry and G. Boudol. The chemical abstract machine. Theoretical Computer Science, 96(1):217-248, 1992.

Non-Patent Document 3

C. A. R. Hoare. Communicating Sequential Processes. Prentice Hall International Series in Computer Science. Prentice Hall, 1985.

Non-Patent Document 4

M. Mejri, B. Ktari, and M. Erhioui. Static analysis on Lyee-oriented software. In Hamido Fujita and Paul Johannesson, editors, New Trends in Software Methodologies, Tools and Techniques, pages 375-394. IOS Press, 2002. Proceedings of 1st International Workshop on Lyee Methodology, Paris.

Non-Patent Document 5

R. Milner. A calculus of communicating systems. Lecture Notes in Computer Science 92, Berlin, 1980. Springer-Verlag.

Non-Patent Document 6

F. Negoro. Principle of Lyee software. 2000 International Conference on Information Society in 21st Century (IS2000), pages 121-189, November 2000.

Non-Patent Document 7

F. Negoro. Introduction to Lyee. The Institute of Computer Based Software Methodology and Technology, Tokyo, Japan, 2001.

Non-Patent Document 8

F. Negoro and I. Hamid. A proposal for intention engineering. 5th East-European Conference Advances in Databases and Information System (ADBIS'2001), September 2000.

Non-Patent Document 9

F. Negoro and I. Hamid. A proposal for intention engineering. International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR 2001), 2001.

Disclosure of the Invention

Problems to be Solved by the Invention

[0004]

The main purpose of this invention is first to formalize, using process algebra, the process of automatic generation of software together with the semantics of Lyee generated software. Actually, the process algebra naturally supports many concepts of the Lyee methodology and consequently formalizes them simply and elegantly. Moreover, a second purpose is to offer the Lyee methodology an abstract machine more suitable than the Von-Newman one. In fact, this new abstract machine considers a program as a chemical solution when molecules (different vectors of the Lyee methodology) interact together to reach a collective goal.

Means for Solving the Problem

[0005]

To solve such problem, the present invention comprises:
a first step for making a statement (defining) a user
requirement to be mounted as one program by a word name, a
definition equation, conditions for executing the definition
equation, input/output attributes, and an attribute of a
value of a word for every logical unit and for every word of
the logical unit; a second step for preparing, from the
statement of the word unit, a signification vector (L_2, L_3,
L_4) and an action vector (I_2, O_4, S_4) formed into a module as
a process cell comprising an input/output channel by Lyee-
calculus; a third step for assembling the signification
vector and the action vector into a set having set conditions
that an interaction is caused by a command from an identical
screen; a fourth step for disposing one control function
module Φ for every set; and a fifth step for disposing a
control function module Ψ being one for every program.

[0006]

A program (software) for producing "software of a
development object", a program generation device, a program
processing device, a tool (including both of tools as the
device and the software), a software development device, a
software development supporting device, or a software
development and management device as one of different
embodyments of the present invention may include: statement
information inserting means for inserting information into an
undefined portion of a signification vector (L_2, L_3, L_4) and

- 4 -

an action vector (I_2, O_4, S_4) formed into a module as a process cell comprising an input/output channel by Lyee-calculus, the information being obtained by forming the user requirement to be mounted as one program into information as a statement inclusing a word name, a definition equation, conditions for executing the definition equation, input/output attributes, and an attribute of a value of a word for every logical unit and for every word of the logical unit; control function Φ disposing means for associating the signification vector and the action vector with a set unit having set conditions that an interaction is caused by a command from from an identical screen so that one control function module Φ controls the set unit; and control function Ψ disposing means for associating one control function module Ψ with the control function Φ so as to control the control function Φ.

[0007]

The present invention is also realized by the software produced by the above-described "method of producing software of the development object", a recording medium on which the software is mounted, or a device (hardware) on which the software is mounted. In this case, the present invention may comprise: a module group obtained by inserting information into an undefined portion of a signification vector (L_2, L_3, L_4) and an action vector (I_2, O_4, S_4) formed into a module as a process cell comprising an input/output channel by Lyee-calculus, the information being obtained by forming the user

requirement to be mounted as one program into information as a statement inclusing a word name, a definition equation, conditions for executing the definition equation, input/output attributes, and an attribute of a value of a word for every logical unit and for every word of the logical unit; one or a plurality of control function modules Φ which associate the module group with a set unit having set conditions that an interaction is caused by a command from an identical screen so as to control the module group by one control function module Φ; and a control function module Ψ which associates a control function Φ so as to control the control function by one control function module Ψ.

[0008]

Furthermore, the present invention is also realized as software which is a model of a software code for use in producing software by the above-described "method of producing the software of the development object", as a recording medium on which the software is mounted, or as a device (hardware) on which the software is mounted. In this case, the present invention may be realized as a model which can be coded as software comprising: a model of a signification vector (L_2, L_3, L_4) and an action vector (I_2, O_4, S_4) formed into a module as a process cell comprising an input/output channel by Lyee-calculus, the model having an undefined portion into which information is to be buried, the information being obtained by forming a user requirement to be mounted as one program into information as a statement

including a word name, a definition equation, conditions for executing the definition equation, input/output attributes, and an attribute of a value of a word for every logical unit and for every word of the logical unit; a model of a control function module Φ having a function of controlling, with one control function module Φ, the signification vector and the action vector into whose undefined portion the information of the statement is inserted, for every a set unit having set conditions that an interaction is caused by a command from an identical screen; and a model of a control function module Ψ having a function of controlling the control function Φ by one control function module Ψ.

[0009]

Furthermore, the present invention can be realized, by the above-described "method of producing the software of the development object", as any of a method of extracting information (document (paper, data)) extracted from a requirement, as the information (document (paper, data)) extracted by such extraction method, as a method of using the extracted information, as an information recording medium on which these information is mounted, as a software in which an information extracting/using method is coded, and as a recording medium/device (hardware) on which the software is mounted. In these cases, the present invention may be realized as information extracted from a software development requirement, comprising: information obtained by forming a user requirement to be mounted as one program into

information as a statement including a word name, a
definition equation, conditions for executing the definition
equation, input/output attributes, and an attribute of a
value of a word for every logical unit and for every word of
the logical unit, the information being to be inserted into
an undefined portion of a model of a signification vector (L_2,
L_3, L_4) and an action vector (I_2, O_4, S_4) formed into a module
as a process cell comprising an input/output channel by Lyee-
calculus; information for associating the signification
vector and the action vector into whose undefined portion the
information of the statement has been inserted with a set
unit having set conditions that an interaction is caused by a
command from an identical screen so as to control the vectors
by one control function module Φ; and information which
associates the control function Φ so as to control the
function by one control function module Ψ.

[0010]

It is to be noted that for the logical unit, Japanese
Patent Application No. 2004-272400 by the same applicant is
referred to, the contents of which are to be incorporated
herein by reference.

Advantages of the Invention

[0011]

A wide range of software problems related to different
fields are efficiently dealt with, and development time,
maintenance time, and documentation volume can be
considerably reduced by using Lyee (as much as about 70 to

80%) as compared with conventional methodologies.

[Best Modes for Carrying Out the Invention]

[0012]

Lyee Methodology: A Formalization using Process Algebra

Abstract: Over the last years, various methodologies and techniques have been elaborated and proposed to improve one or many aspects related to the software development life cycle. However, despite the great effort in this research field, the production of clearly understood and modifiable systems is still an ambitious goal and far from reached. This is due, on one hand, to the complexity and the subtlety of software themselves and, on the other hand, to the limitations of the current methodologies. Recently, a new and very promising methodology, called Lyee, has been proposed. Intended to deal efficiently with a wide range of software problems related to different field, Lyee allows the development of software by simply defining their requirements.

[0013]

Nevertheless, since both the semantics of Lyee generated software together with the process of automatic generation of software from requirements are described using informal language, difficulties and confusions may arise when trying to understand and study this methodology.

[0014]

The main purpose of this invention is first to formalize, using a process algebra, the process of automatic generation of software together with the semantics of Lyee generated

- 9 -

software. Actually, the process algebra naturally supports many concepts of the Lyee methodology and consequently formalizes them simply and elegantly. Moreover, a second purpose is to offer the Lyee methodology an abstract machine more suitable than the Von-Newman one. In fact, this new abstract machine consider a program as chemical solution when molecules (different vectors of the lyee methodology) interact together to reach a collective goal.

1. Introduction

Producing easily and quickly software with high qualities is the basic concern of the software development research field. Over the last years, various methodologies and techniques have been elaborated and proposed to improve one or many aspects related to the software development life cycle. However, despite the great effort in this research field, the production of clearly understood and modifiable systems is still an ambitious goal and far from reached. This is due, on one hand, to the complexity and the subtlety of software themselves and, on the other hand, to the limitations of the current methodologies. In fact, almost all of the proposed methodologies fail to produce clearly understood and modifiable systems and their use is still considered to be an activity accessible only to specialists with a large array of competencies, skills, and knowledge. This, in turn, leads to highly paid personal, high maintenance costs, and extensive checks needing to be

performed on the software. For these reasons, companies are likely to welcome any new methodology promising demonstrable improvement in the software development cycle.

[0015]

Recently, a new and very promising methodology, called Lyee (read as "lee", tail-character word of "governmentaL methodologY for softwarE providencE")(see, Non-Patent Documents 6, 7, 8, and 9), has been proposed. Intended to deal efficiently with a wide range of software problems related to different fields, Lyee allows the development of software by simply defining its requirements. More precisely, a developer has only to provide words, calculation formulae, calculation conditions (preconditions) and layout of screens and printouts, and then leaves in the hands of the computer all subsequent troublesome programming process (e.g., control logic aspects). Despite its recency, the results of the use of Lyee have shown tremendous potential. In fact, compared to conventional methodologies, development time, maintenance time and documentation volume can be considerably reduced by using Lyee (as much as 70 to 80%)(see, Non-Patent Document 7). In Non-Patent Document 4, we proposed some classical static analysis techniques in order to improve many aspects of this methodology, in particular Lyee requirements.

[0016]

Nevertheless, since both the semantics of Lyee generated software together with the process of automatic generation of software from requirements are described using informal

language, difficulties and confusions may arise when trying to understand and study this methodology. In addition, the ideas behind this methodology are described using a sequential language which is not appropriate to support them. In fact, a Lyee generated software is basically made of small components (called vectors in Lyee terminology), where each one has an atomic goal (goal of the vector), that collaborates together by interaction in order to produce the desired results (global goal: the whole goal). It is, however, commonly known that a process algebra which naturally supports concurrency (a plurality of calculation subjects operate while exchanging information with one another) and communication is the intrinsically appropriate language to specify concurrent components and therefore it will provide a formal and worthy foundation to support the Lyee methodology concepts. Among other benefic consequences of using the process algebra in the Lyee methodology, many components (complete vectors or parts of vectors) of the actual version of Lyee will no longer be required. This is because these components, described by the sequential language, have been required by all means for controlling an order of executing a certain group of processes, but their functions are naturally supported by process algebra which supports the process concurrency and communication. Actually, the routing vectors are no longer needed. Thus, we obtain a formal description of Lyee that is simple and that produce program shorter and consuming less time and memory. Besides,

this formal description will be an inevitable start point for many interesting analysis of diverse aspects of this methodology. For instance, to optimize the Lyee generated software or the software generation processes, we need a formal proof that the optimized program is equivalent to its original version. Formal equivalence checking or more generally model checking can be elegantly achieved when using process algebra.

[0017]

In this paper, we first define a formal process algebra, called Lyee-calculus, that easily and naturally supports the basic concepts of the Lyee methodology. In fact, this calculus can be seen as an abstract machine (abstracted concept of a computer) which is more suitable to support the Lyee methodology concepts than the Von Newman one. This machine considers a program as a set of molecules that interact together to produce a final result. Secondly, we show how this calculus can formalize and simplify both the Lyee components (referred to as vectors in Lyee) together with the whole software generation process of the Lyee methodology. Actually, we have formalize the whole process of the automatic generation of Lyee software generation.

[0018]

The remainder of this paper is organized as follows. In Section 2, we define the syntax and the semantics of the Lyee-calculus. In Section 3, we give a short and technical introduction to the Lyee methodology and an overview of its

formalization using the Lyee-calculus. In Section 4, we propose a detailed and complete formalization of the Lyee methodology using Lyee-calculus. In Section 5, we present one case study that concretely shows both how a program is automatically generated from simple requirement together with a step-by-step description of the execution of the generated program. Finally, Section 6 provides concluding remarks on this work, and discusses some future work.

2. Lyee-Calculus

Process algebra is a formal description technique for complex computer systems, especially those involving communicating, concurrently executing components (see, Non-Patent Document 1, 3, 5). Process algebra is generally referred as calculus in the literature since it involves a variety of mathematical and logic concepts (concurrency theory, operational semantics, complexity theory, logic, etc.) that goes beyond the process algebra field.

[0019]

In this section, we give the syntax and the semantics of a new calculus, called Lyee-calculus, that we have specially define to formalize the Lyee methodology.

2.1 Syntax

Lyee-calculus programs are systems composed of independent and parallel processes that communicate using hand-shake technique (a technique to achieve communication

between two processes by setting one process be ready to

receive and the other be ready to send) on named channels.

The channel can be restricted so that only some process can

communicate on them. A process can send a value υ through a

channel ι by doing the action [ι!e] (υ corresponds to the

valuation of e). Similarly, a process can receive a value of

e from a channel ι by doing the action [ι?e].

[0020]

A process can also perform a silent action τ. This

special action is intended to represent an internal behavior

of a system such as synchronization between processes (a

phenomenon in communication between processes, where one

process sends a value or the like and at the same time the

other process receives it). The action is useful to capture

indeterminism in process evolution, that is, in a case where

there are a plurality of process alternatives, and it cannot

be predicted which process is to be executed (refer to the

description of "selection" of Process Syntax Definition 1

described later).

[0021]

The process syntax will be defined as follows.

<Syntax 1>

[0022]

[eq. 1]

$$P,\ Q\ ::=\ [K].P\ |\ (P\ |\ Q)\ |\ P+Q\ |\ P \triangleright Q\ |\ P/L\ |\ .A(\vec{X}) \stackrel{def}{=} P\ |\ \ nil$$

<Syntax 2>

$$K,\ K_1,\ K_2\ ::=\ \{\kappa\}\ |\ K_1 \cup K_2$$

<Syntax 3>

[0023]

[eq. 2]

$$L, L_1, L_2 ::= \quad 0 \mid \{\iota\} \mid L_1 \cup L_2 \mid$$

<Syntax 4>

$\kappa ::= \iota!e \mid \iota?e \mid \tau \mid$

The meanings of symbols for use in the above syntaxes and the intuitive meaning of each syntactic construction are as follows.

<Syntax 1>

[0024]

[Table 1]

Symbol	Definition
P, Q	Processes
::=	The left side of the symbol is defined as the right side
|	Or
[K].P	Sequence
K	Set of actions
P | Q	Parallel composition
P + Q	Choice
P▷Q	Guarded choice
P/ L	Restriction
$A(\vec{X}) \overset{def}{=} P$	Definition
nil	nil process

Syntax 1 has the following intuitive meaning. That is, if there are processes P and Q, relations between them and their contents are defined as any of the followings: sequence ([K].P); parallel composition P | Q); choice (P + Q); guarded

- 16 -

choice (P▷Q); restriction (P/ L); definition; and nil

process (*nil*), where the definition

[0025]

[eq. 3]

$$\underset{=}{def}$$

means that the left side of this symbol is defined by the

right side.

[0026]

Interpretation of each syntactic constuction will be

described later.

<Syntax 2>

[0027]

[Table 2]

Symbol	Definition
K, K_1, K_2	Set of actions
κ	Action
$\{\kappa\}$	Set of a single action (Single action)
$K_1 \cup K_2$	Union of Sets of actions (Set union)

Syntax 2 has the following intuitive meaning. That is,

when there are sets of actions K, K_1, and K_2, those sets are

any of the followings: a set including a single action as an

element ($\{\kappa\}$); a union of sets of actions ($K_1 \cup K_2$).

<Syntax 3>

[0028]

[Table 3]

Symbol	Definition
L, L$_1$, L$_2$	Set of channels
0	Empty set
ι	Channel
{ι}	Set of a single channel (Single channel)
L$_1 \cup$ L$_2$	Union of Sets of channels (Set union)

Syntax 3 has the following intuitive meaning. That is, when there are sets of channels L, L$_1$, and L$_2$, these sets are any of the followings: an empty set (0), a set including only a single channel as an element ({ι}); a union of sets of channels (L$_1 \cup$L$_2$).

<Syntax 4>

[0029]

[Table 4]

Symbol	Definition
κ	Action
ι!e	Send action (send the result of evaluation of the arithmetic expression e on the channel ι)
ι?e	Receive action (receive the value of the arithmetic expression e on the channel ι)
τ	Silent action
$\iota, \jmath$	Channel
e	Arithmetic expression

Syntax 4 has the following intuitive meaning. That is, when there is an action κ, the action is any of the followings: a transmitting action (ι!e) to send a value of a calculation result of an expression e on a channel ι; a receiving action (ι?e) to receive the value of the expression e on the channel ι; and a silent action (τ).

[0030]

The meaning of the syntactic construction defined in Syntax 1 will be described in detail.

1) [K].P: Sequence

Process [K].P , where K is a set of actions, is a process that has to perform all the action in K and then behaves as the process P. The order in which the actions in K have to be executed is not important. Notice that for the sake of simplicity we write $[K_1, \ldots, K_n]$ instead of $[\{K_1, \ldots, K_n\}]$.

2) P|Q: Parallel composition

Process P|Q behaves as processes P and Q running in parallel. Each one of them may interact with the other on channels known to both, or with the outside world (environment or the end-user) independently from the other. When two processes synchronize (sending of a value or the alike from a certain process and receiving of it by the other process occurs at the same time) on the same channel, the whole process will perform an action τ (silent action) and thereafter behave as the remaining process. For instance, let's take the following process:

[0031]

P = P1 | P2

with P1 and P2 defined as follows:

[0032]

[eq. 4]

$$P1 \stackrel{def}{=} [\imath!4].P1'$$
$$P2 \stackrel{def}{=} [\imath?x].P2'$$

The above equation has an intuitive meaning that P1 is a process which transmits a value 4 to the channel $\imath$, and thereafter behaves as process P1'. P2 is a process which waits until a value of an equation x is given to the channel $\imath$, receives the value when given, and thereafter behaves as process P2'.

Then, process P performs the silent action τ at a time when the processes P1 and P2 are synchronized (actually P1 sends the value 4 and simultaneously P2 receives the value), and behaves as the remaining process P1' | P2'. Semantically, we will note this transition as follows:

[0033]

[eq. 5]

$$P \stackrel{\tau}{\to} P1' \mid P2'$$

This equation intuitively means that as a result of an interaction in the process p, the silent action τ is executed, and the whole process shifts to the parallel process P1'|P2'.

[0034]

The relation,

[0035]

[eq. 6]

$$\xrightarrow{\kappa}$$

will be formally defined later within this section.

3) P+Q: Choice

Process P + Q is the process that behaves as Process P or as Q. The choice of the process to be executed is determined deterministically (made by the environment) if both P and Q do not begin with a silent action, otherwise the choice is not determined.

4) P 9 Q: Guarded choice

Process P 9 Q is the process that behaves as P until process Q is activated. Whenever the latter Q is activated, P is stopped and cleared from memory.

5) P/L: Restriction

Process P/L is the process that behaves as P except that it can communicate with the environment by use of channels given to a channel set L only.

6) Symbols of Equation 3: Definition

For example, the symbols of Equation 3 can make definitions represented by the following equation.

[0036]

[eq. 7]

$$A(\vec{X}) \stackrel{def}{=} P$$

wherein A denotes a process identifier,

[0037]

[eq. 8]

$$\vec{X}$$

are variables (parameters) of the Process A, and it is defined that the process of the left side is equal to the process P of the right side. The process P may recursively involve A.

7) nil: nil process

Nil is the process that cannot perform any action. In other words, this is a dead process.

When a certain process ends, the process usually turns to a nil process. When a process ends, thereby resulting in a nil process, the process is not activated any more. Therefore, even if conditions for executing an action are satisfied, any action cannot be executed.

2.2 Modeling of Process to hold Value

In this section, we show how to model a value-holding cell of memory as a process that interacts with its environment through its communication channels. As shown by Fig. 1, we consider that the memory cell has two ports (channels) of communication, *in* and *out*. The basic task of this memory cell is to infinitely wait a value on channel *in* and to make it available on channel *out*. The memory cell may output the same value to the channel *out* as much as necessary until a new value is input into the channel *in*. The channels themselves do not hold any value.

[0038]

We write the process cell $C^x(\upsilon)$, meaning that the memory cell x holds the value υ, as follows:

[0039]

- 22 -

[eq. 9]

$$C^x(v) \stackrel{def}{=} [in^x?y].C^x(y) + [out^x!v].C^x(v)$$

This formal description intuitively means that the process $C^x(v)$ holding the value v behaves as a process $C^x(y)$ (i.e., the held value is changed from v to the value of the equation y), when receiving the value of the equation y in a channel in^x. When the held value v is communicated to a channel out^x, the process behaves as the process $C^x(v)$ while holding the value v even after the transmitting action.

[0040]

By capturing a memory cell as a process holding the value, we can further add intelligence to it. For instance, we can write a cell that does not allow the access to its contents until it will be initialized (initial value is received). This smart cell having an information processing capability can be defined as follows:

[0041]

[eq. 10]

$$Cell(x) \stackrel{def}{=} [in^x?y].C^x(y)$$

A process Cell(x) is a process which behaves as a process $C^x(y)$ holding the value of the equation y, after receiving the value of the equation y in the channel in^x (the process receives the value in the equation y, holds the value of the equation y, and does not transmit any value).

[0042]

Now, giving two processes defined as follows:

[0043]

[eq. 11]

$$P1 \stackrel{def}{=} [in^x!5].nil$$
$$P2 \stackrel{def}{=} [out^x?y].nil$$

The process P1 is a process (process to transmit value 5) which transmits a value 5 to the channel in^x, and turns to nil (ends). The process P2 is a process (process to receive the value in the equation y) which turns to nil (ends) after receiving the value of the equation y in the channel out^x.

It is easy to write a program where these two processes communicate through a cell x (C^x):

[0044]

                P1 | Cell(x) | P2

This program is a program in which the processes P1、 Cell(x)、 and P2 are executed in parallel.

[0045]

Fig. 2 shows interactions among all the above involved processes.

[0046]

Here are the different steps of that program execution:

[0047]

[eq. 12]

$$P1 \mid Cell(x) \mid P2 \;=\; \underline{[in^x!5]}.nil \mid \underline{[in^x?y]}.C^x(y) \mid [out^x?y].nil$$
$$\xrightarrow{\tau}\; \overline{C^x(5)} \mid [out^x?y].nil$$
$$=\; (\,[in^x?y].C^x(y) + \underline{[out^x!5]}.C^x(5)\,) \mid \underline{[out^x?y]}.nil$$
$$\xrightarrow{\tau}\; C^x(5)$$

The above equation intuitively means that the process P1|Cell(x)|P2 is executed through the following first and second steps. An underline indicates a transmitting/receiving action which causes the interaction (synchronization) (the right side denoted with an equal symbol indicates one step).

<First Step>

Three processes are performed in parallel: (1) [in^x!5].nil (transmitting value 5 to channel in^x) ; (2) [in^x?y].C^x(y) (receiving the value to be given to the equation y in the channel in^x, and holding the value of the equation y in a cell x after the receiving) ; and (3) [out^x?y].nil (receiving the value of the equation y in the channel out^x) . As a result, the interaction (synchronization) among the processes occurs: (1) completes the transmission of the value 5 to the channel in^x at the same time (2) receives the value 5 in the channel in^x, and the silent action τ is executed. The process shifts to a parallel process of two processes: (4) C^x(5) (holding the value 5 in the cell x) ; and (3) [out^x?y].nil (receiving the value of the equation y in the channel out^x) .

<Second Step>

From the above definition of the equation 9, (4) C^x(5) can be replaced with a selection process (4)-1

$([in^x?y].C^x(y)+[out^x!5].C^x(5))$. That is, the processes (4)-1 and (3) are performed in parallel. As a result, the interaction (synchronization) occurs: (3) (receiving the value 5 of the equation y in the channel out^x) occurs simultaneously with (4)-1 $[out^x!5].C^x(5)$ (transmitting the value 5 to the channel, and the silent action τ is executed. As a result, the remaining process behaves as $C^x(5)$ (holding the value 5 in the cell x).

2.3 Semantics

Hereafter, we give the formal semantics of the Lyee-calculus. This semantics is defined by the interaction relation,

[0048]

[eq. 13]

$$\xrightarrow{\kappa}$$

At this time, an operator $\rightarrow$ denotes a "relation", and κ denotes an action. The "relation" is a set of a plurality of "relations", and each "relation" which is an element of a set is constituted of the following three constituents. A first constituent is a "process before changed", a second constituent is a "process after changed", and a third constituent is an "action executed during the change".

[0049]

[eq. 14]

$$P \xrightarrow{\kappa} Q$$

wherein the three constituents of the relation → are (P, Q, K).

[0050]

In the above definition, the equation 14 means that there is a reaction amongst the sub-process P such that the whole process can execute the atomic action (a series of undivided actions that does not permit any interrupt, and has to be necessarily completed in order to start the next action separately) κ and becomes the sub-process Q. The interaction relation is an original idea inspired by the Chemical Abstract Machine of Berry and Boudol (see, Non-Patent Document 2). In this model, a process is considered as a chemical solution of molecules waiting to interact.

[0051]

To formally introduce the relation of the equation 13, we need to define the following notions:

[0052]

[eq. 15]

$$\overset{q}{\looparrowright}$$

The above is a symbol for use in simplifying the process, and indicates that the left side is equal to the right side which is a simplified process. The process can be simplified by eliminating the nil process as follows:

[0053]

[eq. 16]

$$P \mid nil \looparrowright P \qquad nil + P \looparrowright P \qquad nil \rhd P \looparrowright P$$
$$nil \mid P \looparrowright P \qquad P + nil \looparrowright P \qquad P \rhd nil \looparrowright P$$

$$nil/L \looparrowright nil$$

[0054]

[eq. 17]

$$[\![e]\!]$$

We denote by [[e]] the set of values containing the result of
the evaluation of the expression e. If e is a variable, a
domain of the value of its evalutaion is the same as that of
the variable (integral, real, etc.). For the sake of
simplicity of the description, we consider that all variables
belong to the real set.

[0055]

[eq. 18]

$$\kappa_{|}$$

is the name of the channel used by the action κ, i.e:

[0056]

[eq. 19]

$$\tau_{|} = \emptyset$$
$$(\imath ! e)_{|} = (\imath ? e)_{|} = \{ \imath \}$$

The equation means that since there does not exist any
channel used by the silent action τ, the set is an empty set

(0). A channel used by a transmitting action (ι!e) and a receiving action (ι?e) is ι.

[0057]

A set of the "relations" (→) has, as elements, the least relation that satisfies 15 rules given by Table 5.

[0058]

Various R symbols within parentheses are symbols indicating rule names. Each rule includes premises and a result, and obtains a result shown in the lower part of a horizontal line at a time when premises in the upper part are satisfied.

[0059]

[Table 5]

$$(R_{\looparrowright}) \frac{P \xrightarrow{\kappa} Q' \quad Q' \looparrowright Q}{P \xrightarrow{\kappa} Q}$$

$$(R_!) \frac{v \in [\![e]\!]}{[\iota!e].P \xrightarrow{\iota!v} P} \qquad (R_?) \frac{v \in [\![e]\!]}{[\iota?e].P \xrightarrow{\iota?v} P[v/e]} \qquad (R_\tau) \frac{v \in [\![e]\!]}{[\tau].P \xrightarrow{\tau} P}$$

$$(R_{[\,]}) \frac{[K_1].P \xrightarrow{\kappa} P'}{[K_1 \cup K_2].P \xrightarrow{\kappa} [K_2]P'} \quad K_2 \neq \emptyset$$

$$(R_+^l) \frac{P \xrightarrow{\kappa} P'}{P+Q \xrightarrow{\kappa} P'} \qquad (R_+^r) \frac{Q \xrightarrow{\kappa} Q'}{P+Q \xrightarrow{\kappa} Q'}$$

$$(R_{\triangleright}^r) \frac{P \xrightarrow{\iota?v} P' \quad Q \xrightarrow{\iota!v} Q'}{P \triangleright Q \xrightarrow{\tau} Q'} \qquad (R_{\triangleright}^l) \frac{P \xrightarrow{\kappa} P'}{P \triangleright Q \xrightarrow{\kappa} P' \triangleright Q} \qquad (R_{\triangleright}^r) \frac{Q \xrightarrow{\kappa} Q'}{P \triangleright Q \xrightarrow{\kappa} Q'}$$

$$(R_{\,|}^\tau) \frac{P \xrightarrow{\iota?v} P' \quad Q \xrightarrow{\iota!v} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad (R_{\,|}^l) \frac{P \xrightarrow{\kappa} P'}{P \mid Q \xrightarrow{\kappa} P' \mid Q} \qquad (R_{\,|}^r) \frac{Q \xrightarrow{\kappa} Q'}{P \mid Q \xrightarrow{\kappa} P \mid Q'}$$

$$(R_=) \frac{P[\vec{Y}/\vec{X}] \xrightarrow{\kappa} P'}{A(\vec{Y}) \xrightarrow{\kappa} P'} \quad A(\vec{X}) = P \qquad (R_/) \frac{P \xrightarrow{\kappa} P'}{P/L \xrightarrow{\kappa} P'/L} \quad \kappa_| \in L$$

For example, the second rule (R₁) means that in a case where the value υ belongs to a set of values of the

- 29 -

expression e (the upper-part premises are satisfied), [ι!e].P, that transmits the value of the expression e to the channel ι and then becomes the process P, shifts to the process P after action of sending the value υ to the channel ι occurs, turns to the process P(result of the lower stage).

3. Informal Formalization of Lyee Requirements

This section presents an overview of how the Lyee methodology generates software from simple user requirements.

Besides, through this overview we progressively introduce how one can simply and efficiently formalize this methodology using the Lyee-calculus. The complete formalization of the process of software generation of the lyee methodology will be given in the next section.

3.1 Lyee requirements

Within the Lyee methodology, requirements are given in a declarative way as a set of statements containing elements such as words together with their definitions, their calculation conditions and their attributes (input/output, types, security attributes, etc.). Table 6 gives an example of Lyee requirements.

[0060]

In the table, "word" indicates a word name, "Definition" indicates a definition expression for generating a value of the word, "Condition" indicates calculation conditions to execute the definition expression, and "IO" indicates

attributes of input/output of the value of the word. "OF" indicates an output to a file, "OS" indicates an output to a screen, "IS" indicates an input from a screen, and IF (not shown) denotes an input from a file. "Type" denotes an attribute of the value, where "int" indicates an integer and "float" indicates a floating-point number. "Security" denotes an attribute relating to security of the value. "secret" indicates non-public level, and "public" indicates public level.

[0061]

[Table 6]

Word	Definition	Condition	IO	Type	Security
		. . .			
a	b + c	b * e > 2	OF	int	secret
c			IS	float	public
b	2 * c + 5	c > 0	OS	float	public
e			IS	float	public
		. . .			

The requirements given in Table 6 correspond intuitively, in a traditional programming language, to the code given in Table 7. S_a denotes a statement of a word a.

[0062]

[Table 7]

Statement	Code

S_a	if b*e>2
	then a:=b+c; output(a);
	end if

S_c	input(c);

S_b	if c>0
	then b:=2*c+5; output(b);
	end if

S_e	input(e);

For example, a code of the statement S_a of the word a means that "if b*e>2 is true, a calculation result of b+c is substituted into the word a, and the value of the word a is output".

[0063]

A code of a statement S_c means "inputting the value into a word c".

[0064]

Within the Lyee methodology, the user does not need to specify the order (control logic) in which these definitions will be executed. As shown in Table 7, despite the fact that the definition of word a uses word b, statement s_b is listed after the statement s_a. As explained in the sequel, from these requirements, and independent of the order of statements, Lyee is able to generate code that computes all the defined words. This simple idea has, as shown in Non-Patent Document 6, 7, 8, and 9, multiple beneficial consequences on the different steps of software development. In fact it allows us to begin developing software even with incomplete requirements. Moreover, the user does not need to

deal with any control logic of the order of executing the words unlike more classical methodologies. The control logic part of the software will be, within the Lyee methodology, automatically generated, reducing consequent programming errors and time. Flexibility is also a major benefit of the Lyee methodology since the maintenance task can be reduced to a simple modification of requirements (add, remove and/or modify words' definitions).

[0065]

Consequently, Lyee systems can be viewed as collections of independent components (statements) that interact together in order to produce the desired result. In this paper, the concepts of our process algebra agree very well with this view. This is because the Lyee-calculus regards a program as a chemical solution of molecules (processes) that mutually interact together to reach the final goal. So, at a first glance, one can view a Lyee requirement LR composed of a set of statements $\{s_1, \ldots, s_n\}$ as a concurrent processes of the Lyee-calculus as follows.

[0066]

$$LR = s_1 | \ldots | s_n$$

This will be intuitively described. A requirement LR is a process in which processes s_1 to S_n are executed in parallel.

3.2 Pallets and Basic Structure (Scenario Function)

From the requirements in Table 6, a program can be

automatically generated that computes the values of a and b and outputs them. This program will simply repeat the execution of these instructions until a fixed point is reached, i.e. any other iteration will not change the value of any word as shown in Fig. 3.

[0067]

From the process algebra view point, the notion of "reaching of a fixed point" is naturally implemented within the semantic. In fact, the processes (molecules) interact together until a state of no possible evolution (fix point) is reached.

[0068]

Let's be more precise about the structure and the content of the program that will be automatically generated by Lyee from requirements. Within the Lyee methodology, the execution of a set of statements, such as the ones given in Table 6, is accomplished in a particular manner. In fact, Lyee distributes the code associated with statements over three spaces, called Pallets (W_{02}, W_{03} and W_{04}) in the Lyee terminology, as shown in Fig. 4.

Pallet W_{02} processes the input words.

Pallet W_{03} computes the calculation conditions of the words and saves the results in boolean variables (having a true/false value). For instance, the calculation condition "b*e>2" used in the definition of the word "a" is calculated in W_{03} and the true/false result is stored in another variable "a_cond".

Pallet W_{04} deals with the calculation of the words by their definition given in the requirements. Pallet W_{04} also outputs the calculated values of the words.

[0069]

Starting from pallet W_{04}, a Lyee program tries to compute the values of all the defined words until a fixed point is reached. Once there is no evolution in W_{04} concerning the computation of the word values, control is given, by routing vector R4C, to pallet W_{02}. In its turn, this second pallet tries repeatedly to input values of words until a fixed point is reached (no other inputs are available) and then, by routing vector R2C, transfer the control to pallet W_{03}. Finally, and similar to pallet W_{04}, pallet W_{03} tries to compute the calculation conditions of the words according to the requirements until a fixed point is reached. As shown in Fig. 5, this whole process ($W_{04} \rightarrow W_{02} \rightarrow W_{03}$) will repeat until a situation of overall stability is reached and the three pallet linked together are called Basic Structure (Scenario Function).

[0070]

Using a sequential language to implement a Lyee requirement forces the author of the Lyee methodology to explicitly specify the order in which the pallet will be executed, the order according to which vectors (modules) in the same pallet have to be executed and how to pass control from one vector to another. In other words, this sequential language forces the introduction of some aspects that do not

belong to fundamental ideas of the Lyee methodology, and that they have considerably complicated this methodology. That is, it is necessary to control repeated execution of the vectors in the pallets, and it is necessary to designate a way, time and the like for transferring the execution from one pallet to another (including transfer between different basic structures). For this purpose, there have been introduced pallet functions (controlling executions in the pallet), routing vectors (determining the pallet to be executed next), and a pallet chain relation (controlling executions between the pallets).

[0071]

However, using the Lyee-calculus, we do not need any more to specify the order according to which the pallets will be executed, the order according to which the vectors of each pallet will be executed, how and when to pass control from one pallet to another, etc. All such details will be, needless to say, naturally managed by the abstract machine of the Lyee-calculus in this paper. Consequently, a Basic Structure (Scenario Function) could be formalized as follows:

[0072]

$$SF = W_{04} \mid W_{03} \mid W_{02}$$

That is, the process of a basic structure SF is a parallel process of processes of three pallets W_{04}, W_{03}, and W_{02}.

[0073]

Notice that the order according to which W_{04}, W_{03} and W_{02} is no more important and the routing vectors that connect

these pallets are also eliminated.

[0074]

Lyee has established a simple program with a fixed structure (called a Predicate Vector in the Lyee terminology) that makes the structure of generated code uniform and independent of the requirement content. A pallet function as global program to control executions of vectors simply calls predicate vectors. The structure of a predicate vector is as shown in Fig. 6.

[0075]

The goal of a predicate vector changes from one pallet to another.

<Pallet W_{04}>

In the pallet W_{04}, the first goal is to give a value to a word according to its definition. The predicate vector that performs this function is referred to as L4. The predicate vectors of the words "a" and "b" in the example shown in Fig. 4, are as shown in Fig. 7.

[0076]

Once there is no evolution in the calculation of the words, the Lyee generated code tries to output the words which will be the next goal. The predicate vector having a goal of outputting values is called as an output action element (output vector: O4).

Additionally, there is a structural action element (referred to as S4) that clears a region.

<Pallet W_{02}>

- 37 -

In pallet W_{02}, we find two predicate vectors having the goals of associating values with input words. The vectors include: an input action vector (referred to as I2) which inputs values to a main memory (outside of the Lyee program region); and L2 which checks attribute of a value and inputs it into the Lyee program region.

<Pallet W_{03}>

In pallet W_{03}, The goal of the predicate vector L3 is to compute a condition for executing the definition (i.e., L4) of the word, which are specified within requirements as shown in Fig. 8.

[0077]

Finally, the Lyee program associated with the requirements given in Table 6 is as shown in Table 8.

[0078]

One pallet function disposed in each pallet calls (activates) the predicate vectors in the pallet. The pallet functions are called by one pallet chain function disposed in the program. The pallet chain function calls the corresponding pallet function in accordance with designations by R4, R2, and R3 of the pallets. Meanings of other vector programs are as shown in comments in Table 8. A processing object of each predicate vector L4, L2, and L3 (referred to generically as a signification vector) is a word, and a processing object of the input/output vector (I2 and O4) and the structural vector (S4) is a set of words.

[0079]

[Table 8]

Pallet	Program	Comments
W_{04}	Call S4	Initialize memory area
	Do	
	Call L4 a	Calculate the definition of the word a
	Call L4 b	Calculate the definition of the word b
	while a fixed point is not reached	Repeat until a fixed point is reached
	Call O4	Output the results
	Call R4	Designate W_{02} as the next route
W_{02}	Call I2	Input (to the memory)
	Do	
	Call L2 e	Input the word e
	Call L2 c	Input the word c
	while a fixed point is not reached	Repeat until a fixed point is reached
	Call R2	Designate W_{03} as the next route
W_{03}	Do	
	Call L3 a	Calculate a_cond (the condition of the word a)
	Call L3 b	Calculate b_cond (the condition of the word b)
	while a fixed point is not reached	Repeat until a fixed point is reached
	Call R3	Designate W_{04} as the next route

Now, using the Lyee-calculus, W_{04}, W_{03} and W_{02} can be specified as follows:

[0080]

[eq. 20]

$$W_{04} = S4 \mid L4\_a \mid L4\_b \mid O4$$

$$W_{03} = L3\_a \mid L3\_b$$

$$W_{02} = L2\_e \mid L2\_c \mid I2$$

The W_{04} process is a parallel process of processes S4、 L4_a、 L4_b, and O4.

The W_{03} process is a parallel process of processes L3_a

- 39 -

and L3_b.

The W_{02} process is a parallel process of processes L2_e、
L2_c、 and I2.

The formal definition of the different vectors (L2, L3,
L4, I2, O4, S4, etc.) will be given in the next section.

3.3 Process Route Diagrams

The Scenario Function presented in the previous section
can be a complete program for a simple case of given
requirements, particularly when all the input and output
words belong to the same screen and there is no use of any
database. If we need to input and output words that belong to
databases or to different screens interconnected together,
then the situation will be more complicated. For the sake of
simplicity, we deal, in the sequel, only with the case when
we have many screens. Suppose for instance that we have three
interconnected screens, as shown in Fig. 9, allowing a user
to navigate from one to another. Suppose in each one of them
the user can input, compute and output words. Therefore, in
the specification, the user must specify how these screens
are interconnected.

[0081]

Furthermore, it is not convenient to define only one
scenario function in which we compute all the words defined
in all the screens. This is because in fact, some screens
may not be visited for a given execution of the program and
then the computation of the value of their words that have

not been visited will be lost. For that reason, Lyee

associates with each screen its owner scenario function that

will be executed only if this screen is visited. The scenario

functions associated with screens are connected together

showing the move from one of them to another. In the Lyee

terminology, many scenario functions connected together make

up a Process Route Diagram as shown in Fig. 10.

[0082]

Using the Lyee-calculus, we can formalize a screen s_k as

a process $\Phi(s_k)$ that control the execution of the process

SF(s_k) as follows:

[0083]

[eq. 21]

$$\Phi(s_k) = SF(s_k) \triangleright \left[\imath^{s_k}?1 \right].\Phi(s_k)$$

That is, the process $\Phi(s_k)$ behaves as the process SF(s_k)

(process having a basic structure of the processing relating

to a screen s_k). When the process receives a true value in a

channel $\imath^{sk}$, the process stops SF(S_k), and behaves as the

process $\Phi(s_k)$. In actual movement, the control process $\Phi(s_k)$

actives the process SF(s_k). When the control process $\Phi(s_k)$

receives a signal on the channel $\imath^{sk}$ associated to the screen

s_k (i.e., when the screen s_k is reactivated), the control

process itself is reactivated. Therefore, when the control

process $\Phi(s_k)$ is reactivated, SF(S_k) once ends, and is

reactivated, thereby returning to a fresh instance.

[0084]

A concrete example will be given later in this paper to show how the function $\Phi(s_k)$ works.

3.4 Lyee Program

To summarize, a Lyee program by the Lyee methodology has the following structure. The Lyee program contains several process route diagrams (PRD). Each of them is a set of interconnected scenario functions (SF). Each scenario function contains three interconnected pallets W_{02}, W_{03} and W_{04}. Finally, the pallet contains predicate vectors.

Since the predicate vectors are modules of minimum units of the Lyee program, the predicate vectors are also referred to as atomic vectors. The predicate vectors include signification vectors (generic term of L2, L3, and L4) and action vectors (generic term of I2, O4, S4, R2, R3, and R4). As a control module which controls the execution of these predicate vectors, each pallet is provided with a pallet function. As a module which controls the execution of the pallet function, a program is provided with one pallet chain function.

[0085]

Using the Lyee-calculus, a Lyee program P containing the screens s_1, . . . , s_k, is formalized as follows:

[0086]

[eq. 22]

$$\mathcal{P}(s_1, \ldots, s_k) = \Psi(s_1, \ldots, s_k)/\mathcal{L}(s_1, \ldots, s_k)$$

where $L(s_1, \ldots, s_k)$ is the set of input and output channels. This restricts $(s_1, \ldots, s_k)$ to communicate with the environment only on channels in $L(s_1, \ldots, s_k)$.

[0087]

The function Ψ is defined as follows:

[0088]

[eq. 23]

$$\Psi(s_1, \ldots, s_k) = \left(\big|_{s \in \{s_1, \ldots, s_k\}} \left[\iota^s ? 1 \right].\Phi(s) \right) \triangleright \left[\iota^{s_0} ? 1 \right].nil$$

This function Ψ activates a process $\Phi(s)$ in order to activate $SF(s)$ corresponding to a screen s_k whenever a true value is received on the corresponding channel ι^s corresponding to the screen s (s is one of screens s_1, ..., s_k). The channel ι^s formalizes a button or a menu item that a user may use to activate the corresponding screen. When the function Ψ receives a true value on a channel ι^{s_0}, the function Ψ kills all other processes, and the function itself also ends (turns to nil). The channel ι^{s_0} formalizes the Exit button (or a corresponding menu item).

[0089]

Consequently, a Lyee system can be viewed as a collection of independent concurrent processes that communicate with each other to compute the desired output words. In contrast with conventional sequential view of Lyee methodology, there is no need for all routing vectors. The role of the control function is largely simplified. There is no need of working memory areas, etc. More details about

$\Psi(s_1, \ldots, s_k)$ together with concrete examples will be given in the sequel.

[0090]

The next section gives a more detailed and complete formalization of the Lyee methodology.

4. Formalization of the Lyee Methodology

There will be described how to automatically generate software from a simple user requirement by use of the Lyee-calculus.

It is assumed that the user requirement includes k screens $\{s_1, \ldots, s_k\}$. It is also assumed that each screen includes a set of statements. In this case, each statement has a format (w, e, c, InOut, type), wherein: w denotes a word name; e denotes a definition expression; c denotes a condition for execution of the definition expression; InOut designates whether the word is input, output, both, or neither (the value i is used for input, the value o is for output, the value io is for both input and output, and an empty field is used for neither input nor output); and type denotes an attribute (type) of a value (e.g., type B is assigned as a symbol indicating a button).

To define the program P $(s_1, \ldots, s_k)$ on this requirement, signification and action vectors (processes) are formalized as shown in the following table.

[0091]

First, undefined symbols used in the formalization will

be described.

Let Use(e) denotes the set of words used in the expression e. For instance, Use(a * b + 1) ={a, b}. Also, F(S) is a function which takes a set of input values for variable S and which returns a set of receiving actions to input those values.

[0092]

[eq. 24]

$$\mathcal{F}(\emptyset) = \emptyset$$
$$\mathcal{F}(\{x\} \cup A) = \{\imath_4^x?x\} \cup \mathcal{F}(A)$$

The above has the following meaning:

When there is not any input value ((F(0)), there is not any receiving action (0 is an empty set).

A receiving action (F({x}∪A)) having input values x and A is a union set of receiving actions, one of which is to receive a value of x on the channel of

[0093]

[eq. 25]

$$\imath_4^x$$

and the other is a receiving action F(A) to receive input A.

[0094]

The signification vectors are shown in Table 9.

[0095]

[Table 9]

Vector	Comments
$L_4 \ (x, e) \ =$	<L_4 of Word x which Definition is e (InOut attribute of its Statement is 'o') >
$[z_3^x?1]$.	Wait to receive the value true (1) on the channel z_3^x (from L_3)
$[\mathcal{F}(Use(e))]$.	Wait to receive the vale of all the Use of e form the memory
$[j_4^x!e]$.	Evaluate e and put the result in the memory cell of x
nil	finish
$L_3 \ (x, c) \ =$	<L_3 of Word x which Condition is c (InOut attribute of its Statement is 'o' or 'io') >
$[\mathcal{F}(Use(c))]$.	Wait to receive the value of all the Use of c from the memory
$[z_3^x!c]$.	evaluate c (its value is Boolean) and send the result on the channel z_3^x (to L_4)
nil	finish
$L_2 \ (x) \ =$	<L_2 of Word x (InOut attribute of its Statement is 'i') >
$[z_2^x?x]$.	Wait to receive the value of x on the channel z_2^x (from the input vector)
$[j_4^x!x]$.	Send the value of x on the channel of j_4^x (save the value of x in the memory cell of x)
nil	finish

The action vectors are shown in Table 10.

[0096]

[Table 10]

Input Vector	Comments
$I_2 (x) =$ $[d_x?x].$ $[v_2^x!x].$ nil	$< I_2$ of input words group which element is $\{x\}$ $>$ Wait to receive a value for Word x on the input channel d_x Send the value of x on the channel v_2^x (to L_2) finish

Output Vector	Comments
$O_4 (x) =$ $[v_4^x?x].$ $[d_x!x].$ nil	$< O_4$ of output words group having the element $\{x\}$ $>$ Wait to receive the value of x from the memory Send the value of x on the output channel d_x finish

Structural Vector	Comments
$S_4 (x) =$ $[j_4^x?y].S_4^x(y)$ $S_4^x(y) =$ $[j_4^x?z].S_4^x(z)+$ $[v_4^x!y].S_4^x(y)$	$<S_4$ of Word $x>$ Wait to receive any value. Cannot send any value until you receive any value. Once receive a value (it is called 'initialization'), save the value and behave as $S_4^x(y)$. If you receive a new value z, behave as $S_4^x(z)$ (change the content of the cell x). If you send y which has been saved, behave as $S_4^x(y)$ (do not change the content of the cell x). In other word, once you receive and save a value, you can send it other than receiving and saving.

Routing Vector	Comments
$R_3(b, e, s) =$ $[d_b?clik].$ $[\mathcal{F}(Use(e))].$ $[v^s!e].$ nil	$< R_3$ of Word b (Button b)$>$ Wait until the button b be pushed Evaluate the condition of the button b Send the result to the screen s (in other words, activate s if $e = 1$(true) finish

Among the vectors, S_4 is a memory cell which holds a value as described in 2.2, and it is a smart cell having an

ability to "prevent the content from being accessed until the value is initialized (an initial value is received)".

[0097]

A routing vector is a naturally unnecessary process in the Lyee-calculus, but the element is described in order to show a way of formalization by the Lyee-calculus.

Pallet:

The three pallets W_{02}, W_{03} and W_{04} of a given screen s are formalized as follows:

[0098]

[eq. 26]

$$
\begin{aligned}
W_{02}(s) &= |_{(w,*,*,i,\overline{B})\in s}\, I_2(w)\, |_{(w,*,*,i,\overline{B})\in s}\, L_2(w) \\
W_{03}(s) &= |_{(w,*,c,*,\overline{B})\in s}\, L_3(w,c)\, |_{(w,e,c,i,*,B)\in s}\, R_3(w,c,e) \\
W_{04}(s) &= |_{(w,*,*,*,\overline{B})\in s}\, S_4(w)\, |_{(w,e,*,*,\overline{B})\in s} L_4(w,e))\, |_{(w,*,*,o,*,*)\in s}\, O_4(w)
\end{aligned}
$$

where

[0099]

[eq. 27]

$$\overline{B}$$

denotes any type except B (B denotes the button), that means complementary of B, and * denotes anything.

Scenario Function:

The scenario function, SF(s)of the screen s is formalized as follows:

[0100]

[eq. 28]

$$SF(s) = W_{04}(s) \mid W_{03}(s) \mid W_{02}(s)$$

Control Function:

The control function attached to a screen s, is formalized as follows:

[0101]

[eq. 29]

$$\Phi(s) = SF(s) \triangleright [\imath^{s_k}?1].\Phi(s)$$

The control function attached to a set of screens is formalized as follows:

[0102]

[eq. 30]

$$\Psi(s_1, \ldots, s_k) = \left(\big|_{s \in \{s_1, \ldots, s_k\}} [\imath^{s}?1.]\Phi(s) \right) \triangleright [\imath^{s_0}?1].nil$$

We suppose that s_0 is the screen that we find when we exit the program (exit screen that does not belong to the screens of the program itself).

[0103]

Here, roles of the control function are summarized as follows.

[0104]

[Table 11]

Control Function		function
Ψ	Place one for each program	· It is activated by the end-user and is killed when it receives the true value from Finish button being pushed. · When it receives the value true from a button pushed by the end-user, it activates the Φ corresponding to the button. · When it receives the value true from Finish button, it kills all Φs before kills itself.
Φ	Place one for each SF	· It is activated and killed by Ψ. · When it receives the value true on the designated channels from a button being pushed by the end-use, it activates the corresponding SF.

In this paper, as one of methods of realizing the Lyee-calculus, a single SF is disposed in a screen, and such single SF is formalized so as to contain all processes of input/output and calculation relating to the corresponding screen. Therefore, a single control function Φ of the SF is disposed to a single screen. However, it is also a possible alternative to formalize the screen so that a plurality of SFs and control functions Φ are disposed to a single screen. Implementation methods may be determined by efficiency of the program.

Lyee Program:

Finally, the Lyee program $P(s_1, \ldots, s_k)$ associated with the requirement containing the screens $s_1, \ldots, s_k$ is as follows:

[0105]

[eq. 31]

$$\mathcal{P}(s_1, \ldots, s_k) = \Psi(s_1, \ldots, s_k)/\mathcal{L}(s_1, \ldots, s_k)$$

where the set $L(s_1, \ldots, s_k)$ contains all the input and output channels and it is defined as follows:

[0106]

[eq. 32]

$$\mathcal{L}(s_1, \ldots, s_k) = (\bigcup_{(w,*,*,i/o,*) \in s_k} \{d_w\}) \cup \{i^{s_1}\}$$

The meaning of i/o is that this field has to contain the value i, o or io. We suppose also that s_1 is the first screen that appears when the user runs this program.

[Example 1]

[0107]

5. Case Study 1

In this section, we give a concrete example and describe step by step how a Lyee program progress in order to compute words within our Lyee-calculus.

[0108]

The example that we present contains only one screen (another example with two screens is given in the appendix). As shown by the requirement given in Table 12, a user can input a word a, waits for the value of b and then exit the screen by pushing the button B_0. Fig. 11 illustrates such a screen.

[0109]

[Table 12]

Word	Definition	Condition	IO	Type
a			i	real
b	2*a	a>0	o	real
B_0	s_0	Click	i	B

This screen s_1 is composed from three statements:

[0110]

[eq. 33]

$$s_1 = \{(a,,,i,real),(b,2*a,a>0,o,real),(B_0,s_0,Click,B)\}$$

According to the definition of a generic Lyee program given in the previous section, the Lyee program associated to the requirement given by Table 12 is as follows:

[0111]

[eq. 34]

$$SF(s_1) = W_{04}(s_1) \mid W_{03}(s_1) \mid W_{02}(s_1)$$

$$\Phi(s_1) = SF(s_1) \rhd [\imath^{s_1}?1].\Phi(s_1)$$

$$\Psi(s_1) = ([\imath^{s_1}?1].\Phi(s_1)) \rhd [\imath^{s_0}?1].nil$$

$$\mathcal{L}(s_1) = \{d_a, d_b, d_{B_0}, \imath^{s_1}\}$$

$$\mathcal{P}(s_1) = \Psi(s_1)/\mathcal{L}(s_1)$$

wherein $W_{02}(s_1)$, $W_{03}(s_1)$ and $W_{04}(s_1)$ are as shown hereafter. As described above, since the process of R_3 is not essential in the Lyee-calculus, it can be omitted.

[0112]

[Table 13]

$W_{02}(s_1) = L_2(a) \mid I_2(a)$	
$L_2(a) = [\imath_2^a?a].$ $[\jmath_4^a!a].$ nil	$I_2(a) = [d_a?a].$ $[\imath_2^a!a].$ nil

$W_{03}(s_1) = L_3(b, a > 0) \mid R_3(B_0, click, s_0)$	
$L_3(b, a > 0) = [\imath_4^a?a].$ $[\imath_3^b!(a > 0)].$ nil	$R_3(B_0, click, s_0) = [d_{B_0}?click].$ $[\imath^{s_0}!1].$ nil

$W_{04}(s_1) = S_4(a) \mid S_4(b) \mid L_4(b, 2 * a) \mid O_4(b)$			
$S_4(a) = [\jmath_4^a?a].$ $S_4^a(a)$ $S_4(b) = [\jmath_4^b?b].$ $S_4^b(b)$	$L_4(b, 2 * a) = [\imath_3^b?1].$ $[\imath_4^a?a].$ $[\jmath_4^b!(2 * a)].$ nil	$O_4(b) = [\imath_4^b?b].$ $[d_b!b].$ nil	

Suppose now, that the end-user (environment) of this program P(S₁) wants to perform the following sequence of actions : runs the program (activates the screen s₁), gives the value 7 to the word a, waits for the value of the word b and then exits the program by pushing the button B₀. This end-user's comportment can be captured by the following process ε:

[0113]

[Table 14]

Process	Comments
$\mathcal{E} = [\imath^{s_1}!1].\mathcal{E}_1$	Activate the program (screen s₁)
$\mathcal{E}_1 = [d_a!7].\mathcal{E}_2$	Give the value 7 to the word a
$\mathcal{E}_2 = [d_b?x].\mathcal{E}_3$	Get the value of the word b
$\mathcal{E}_3 = [d_{B_0}!click].nil$	Click the button B₀ to exit

Fig. 12 shows the process of the program $P(S_1)$ and the end-user's process by the Lyee-calculus described above.

[0114]

Let's see now how the program $P(S_1)$ interacts with this environment to reach the desired goal. In other words, we trace the program $P(S_1)$ progress when executed in parallel with the program ε that incorporates the end-user actions, i.e., we look for the behaviors of $P(s_1) \mid \varepsilon$. The steps of this process are as shown hereafter. In the process definition, comments are added with definitions which account for grounds on which the left side of an equation mark is replaced with the right side. A comment is also added to a portion where synchronization (transmission and reception of the value are simultaneously established on the same channel) occurs by the interaction between the processes. An underlined portion shows a synchronized transmission and reception actions.

[0115]

[Table 15]

$\mathcal{P}(s_1) \mid \mathcal{E}$

$=$ ⟨By definition of P and ε⟩

$\Psi(s_1)/\mathcal{L}(s_1) \mid [\imath^{s_1}!1].\mathcal{E}_1$

$=$ ⟨By definition of Ψ⟩

$(([\imath^{s_1}?1].\Phi(s_1)) \rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1) \mid [\imath^{s_1}!1].\mathcal{E}_1$

$\xrightarrow{\tau}$ ⟨Synchronization−End-user activates the program by running the screen s_1⟩

$(\Phi(s_1) \rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1) \mid \mathcal{E}_1$

$=$ ⟨By definition of Φ and ε_1⟩

$((SF(s_1) \rhd [\imath^{s_1}?1].\Phi(s_1)) \rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1) \mid [d_a!7].\mathcal{E}_2$

$=$ ⟨By definition of S F⟩

$(((W_{04}(s_1) \mid W_{02}(s_1) \mid W_{03}(s_1)) \rhd [\imath^{s_1}?1].\Phi(s_1)) \rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1) \mid [d_a!7].\mathcal{E}_2$

$=$ ⟨By definition of W_{02} and consequently I_2⟩

$(((W_{04}(s_1) \mid L_2(a) \mid ([d_a?a].[\imath^a_2!a].nil) \mid W_{03}(s_1)) \rhd \ldots) \rhd \ldots)/\mathcal{L}(s_1) \mid [d_a!7].\mathcal{E}_2$

$\xrightarrow{\tau}$ ⟨Synchronization−Through I_2 end-user gives the value 7 to the word a⟩

$(((W_{04}(s_1) \mid L_2(a) \mid ([\imath^a_2!7].nil) \mid W_{03}(s_1)) \rhd \ldots) \rhd \ldots)/\mathcal{L}(s_1) \mid \mathcal{E}_2$

$=$ ⟨By definition of L_2⟩

$(((W_{04}(s_1) \mid ([\imath^a_2?a].[\jmath^a_4!a].nil) \mid ([\imath^a_2!7].nil) \mid W_{03}(s_1)) \rhd \ldots) \rhd \ldots)/\mathcal{L}(s_1) \mid \mathcal{E}_2$

$\xrightarrow{\tau}$ ⟨Synchronization−The value 7 of a is communicated to $L_2(a)$⟩

$(((W_{04}(s_1) \mid ([\jmath^a_4!7].nil) \mid W_{03}(s_1)) \rhd \ldots) \rhd \ldots)/\mathcal{L}(s_1) \mid \mathcal{E}_2$

$=$ ⟨By definition of W_{04}⟩

$((S_4(a) \mid S_4(b) \mid L_4(b, 2 * a) \mid O_4(b) \mid ([\jmath^a_4!7].nil) \mid W_{03}(s_1)) \rhd \ldots)/\mathcal{L}(s_1) \mid \mathcal{E}_2$

$=$ ⟨By definition of $S_4(a)$⟩

$((([\jmath^a_4?a].S^a_4(a)) \mid S_4(b) \mid L_4(b, 2 * a) \mid O_4(b) \mid ([\jmath^a_4!7].nil) \mid W_{03}(s_1)) \rhd \ldots$

$\xrightarrow{\tau}$ ⟨Synchronization−The value 7 of a is sent from $L_2(a)$ and stored in the memory⟩

$((S^a_4(7) \mid S_4(b) \mid L_4(b, 2 * a) \mid O_4(b) \mid W_{03}(s_1)) \rhd \ldots$

$=$ ⟨By definition of $S^a_4(7)$ and W_{03}⟩

$((([\jmath^a_4?z].S^a_4(z) + [\imath^a_4!7].S^a_4(7)) \mid \ldots \mid (L_3(b, a > 0) \mid R_3(B_0, click, s_0))) \rhd \ldots$

$=$ ⟨By definition of L_3⟩

$((([\jmath^a_4?z].S^a_4(z) + [\imath^a_4!7].S^a_4(7)) \mid \ldots \mid (([\imath^a_3?a].[\imath^b_3!(a > 0)].nil) \mid \ldots$

$\xrightarrow{\tau}$ ⟨Synchronization−The value of a is communicated to $L_3(b)$ to compute the b condition⟩

$((S^a_4(7) \mid S_4(b) \mid L_4(b, 2 * a) \mid O_4(b) \mid ([\imath^b_3!(7 > 0)].nil) \mid \ldots$

$=$ ⟨By definition of L_4 and evaluation of $(7 > 0)$⟩

$((S^a_4(7) \mid S_4(b) \mid ([\imath^b_3?1].[\imath^a_4?a].[\jmath^b_4!(2 * a)].nil) \mid O_4(b) \mid ([\imath^b_3!1].nil) \mid \ldots$

[0116]

[Table 16]

$\xrightarrow{\tau}$ ⟨Synchronization – The b condition is evaluated and communicated to

$L_4(b)$ to activate the computation of b ⟩

$((S_4^a(7) \mid S_4(b) \mid ([i_4^a?a].[j_4^b!(2*a)].nil) \mid O_4(b) \mid \ldots$

= ⟨By definition of $S_4^a(7)$⟩

$((([j_4^a?z].S_4^a(z) + [i_4^a!7].S_4^a(7)) \mid S_4(b) \mid ([i_4^a?a].[j_4^b!(2*a)].nil) \mid O_4(b) \mid \ldots$

$\xrightarrow{\tau}$ ⟨Synchronization – The value of a is communicated to L_4⟩

$((S_4^a(7) \mid S_4(b) \mid ([j_4^b!(2*7)].nil) \mid O_4(b) \mid R_3(B_0, click, s_0)) \rhd \ldots$

= ⟨By definition of $S_4(b)$ and evaluation of $(2*7)$⟩

$((S_4^a(7) \mid ([j_4^b?b].S_4^b(b)) \mid ([j_4^b!(14)].nil) \mid O_4(b) \mid R_3(B_0, click, s_0)) \rhd \ldots$

$\xrightarrow{\tau}$ ⟨Synchronization – The value of b (14) is sent from L_4(b) stored in

the memory⟩

$((S_4^a(7) \mid S_4^b(14) \mid O_4(b) \mid R_3(B_0, click, s_0)) \rhd \ldots$

= ⟨By the definition of $S_4^b(14)$ and $O_4(b)$⟩

$((S_4^a(7) \mid ([j_4^b?z].S_4^b(z) + [i_4^b!14].S_4^b(14)) \mid ([i_4^b?b].[d_b!b].nil) \mid R_3(B_0, click, s_0)) \rhd \ldots$

$\xrightarrow{\tau}$ ⟨Synchronization – The value of b (14) is communicated to O_4⟩

$((S_4^a(7) \mid S_4^b(14) \mid ([d_b!14].nil) \mid R_3(B_0, click, s_0)) \rhd \ldots)/\mathcal{L}(s_1) \mid \mathcal{E}_2$

= ⟨By definition of ε_2⟩

$((S_4^a(7) \mid S_4^b(14) \mid ([d_b!14].nil) \mid R_3(B_0, click, s_0)) \rhd \ldots)/\mathcal{L}(s_1) \mid [d_b?x].\mathcal{E}_3$

$\xrightarrow{\tau}$ ⟨Synchronization – The b value, stored in O_4, is communicated to the
environment⟩

$((S_4^a(7) \mid S_4^b(14) \mid R_3(B_0, click, s_0)) \rhd \ldots)/\mathcal{L}(s_1) \mid \mathcal{E}_3$

= ⟨By definition of R_3 and ε_3⟩

$((S_4^a(7) \mid S_4^b(14) \mid ([d_{B_0}?click].[i^{s_0}!1].nil)) \rhd \ldots)/\mathcal{L}(s_1) \mid [d_{B_0}!click].nil$

$\xrightarrow{\tau}$ ⟨Synchronization – The end-user push button B_0 to exit⟩

$((S_4^a(7) \mid S_4^b(14) \mid ([i^{s_0}!1].nil)) \rhd [i^{s_1}?1].\Phi(s_1)) \rhd [i^{s_0}?1].nil)/\mathcal{L}(s_1)$

$\xrightarrow{\tau}$ ⟨Synchronization – The system ends its execution⟩

nil

To summarize, here are the main steps of the program progress:

The end-user sends a command for activating the program to thereby activate a control function Ψ (s_1). The control function Ψ (s_1) activates a control function Φ (s_1). The control function (s_1) activates an initial screen s_1 and SF (s_1), whereby the program is activated.

[0117]

[eq. 35]

$$\left(\left(\underline{[z^{s_1}?1]}.\Phi(s_1) \right) \triangleright [z^{s_0}?1].nil \right) / \mathcal{L}(s_1) \mid \underline{[z^{s_1}!1]}.\mathcal{E}_1$$

When the end-user gives (transmits) a value "7" to the word a, I_2(a) in SF(s_1) receives the value 7.

[0118]

[eq. 36]

$$\left(\left((W_{04}(s_1) \mid L_2(a) \mid (\underline{[d_a?a]}.[z^a_2!a].nil) \mid W_{03}(s_1)) \triangleright \dots \right) \triangleright \dots \right) / \mathcal{L}(s_1) \mid \underline{[d_a!7]}.\mathcal{E}_2$$

The value 7 of the word a is communicated to L_2(a) by I_2(a).

[0119]

[eq. 37]

$$\left(\left((W_{04}(s_1) \mid (\underline{[z^a_2?a]}.[j^a_4!a].nil) \mid (\underline{[z^a_2!7]}.nil) \mid W_{03}(s_1)) \triangleright \dots \right) \triangleright \dots \right) / \mathcal{L}(s_1) \mid \mathcal{E}_2$$

The value 7 of the word a communicated by L_2(a) is stored in the memory, i.e., S_4(a) (the initial value is stored in S_4(a), and the memory is initialized).

[0120]

[eq. 38]

$$\left(((\underline{[j^a_4?a]}.S^a_4(a)) \mid S_4(b) \mid L_4(b, 2*a) \mid O_4(b) \mid (\underline{[j^a_4!7]}.nil) \mid W_{03}(s_1)) \triangleright \dots \right.$$

The value 7 of a is communicated from the initialized

memory S_4 of a to L_3 (b, a>0) to compute the calculation

conditions of b.

[0121]

[eq. 39]

$$\Big(\big(([j_4^a?z].S_4^a(z) + [\imath_4^a!7].S_4^a(7)) \mid \dots \mid (([\imath_4^a?a].[\imath_3^b!(a > 0)].nil) \mid \dots$$

The calculation conditions of b are evaluated (a > 0 is

judged to be true (1)), and a value of 1 is communicated to

L_4 (b, 2*a).

[0122]

[eq. 40]

$$\Big((S_4^a(7) \mid S_4(b) \mid ([\imath_3^b?1].[\imath_4^a?a].[j_4^b!(2 * a)].nil) \mid O_4(b) \mid ([\imath_3^b!1].nil) \mid \dots$$

The value 7 of a is communicated to L_4 (b, 2*a) from the

memory S_4 which holds the value 7.

[0123]

[eq. 41]

$$\Big((([j_4^a?z].S_4^a(z) + [\imath_4^a!7].S_4^a(7)) \mid S_4(b) \mid ([\imath_4^a?a].[j_4^b!(2 * a)].nil) \mid O_4(b) \mid \dots$$

A definition expression "2*a" of b is calculated, and a

value 14 of b is stored in the memory, i.e., S_4(b) (S_4(b) is

initialized).

[0124]

[eq. 42]

$$\Big((S_4^a(7) \mid ([j_4^b?b].S_4^b(b)) \mid ([j_4^b!(14)].nil) \mid O_4(b) \mid R_3(B_0, click, s_0)) \triangleright \dots$$

The value 14 of b is communicated from the memory S_4 to

the output process O_4(b).

[0125]

[eq. 43]

$$\left(\left(S_4^a(7) \mid ([j_4^b?z].S_4^b(z) + \underline{[\iota_4^b!14]}.S_4^b(14)) \mid (\underline{[\iota_4^b?b]}.[d_b!b].nil) \mid R_3(B_0, click, s_0) \right) \rhd \ldots \right.$$

10. The value of b is communicated to the environment by O_4, and the end-user receives the value of b. At this time, processes belonging to $SF(s_1)$ being activated are S_4 which holds values of a and b, and $R3(B_0,$ click, $s_0)$.

[0126]

[eq. 44]

$$\left(\left(S_4^a(7) \mid S_4^b(14) \mid (\underline{[d_b!14]}.nil) \mid R_3(B_0, click, s_0) \right) \rhd \ldots \right)/\mathcal{L}(s_1) \mid \underline{[d_b?x]}.\mathcal{E}_3$$

11. The end-user pushes button B_0 to exit, and $R3(B_0,$ click, $s_0)$ receives the value click, and evaluates its own calculation conditions to transmit a true value of 1 to a control function $\psi(S_1)$. The control function $\psi(S_1)$ which has received the true value of 1 ends $\Phi(S_1)$ before the function itself ends, and $\Phi(S_1)$ ends $SF(S_1)$ before $\Phi(S_1)$ itself ends.

[0127]

[eq. 45]

$$\left(\left(S_4^a(7) \mid S_4^b(\overline{14}) \mid (\underline{[d_{B_0}?click]}.[\iota^{s_0}!1].nil) \right) \rhd \ldots \right)/\mathcal{L}(s_1) \mid \underline{[d_{B_0}!click]}.nil$$

Fig. 13 illustrates each of these steps.

[0128]

The parenthesized numerals correspond to the above description of each step. The numerals that do not have any parentheses indicate values to be transmitted/received.

[0129]

Furthermore, each transition could be validated by a proof. For instance, there is the transition of the

following process (corresponding to the step 1). The synchronization occurs between P(s₁) and ε (transmission and reception of the program activation command), and process transition is represented such that the process shifts to a parallel process of P(s₁) and ε₁.

[0130]

[eq. 46]

$$\left(\mathcal{P}(s_1) \mid \mathcal{E}\right) \xrightarrow{\tau} \left(\mathcal{P}_1(s_1) \mid \mathcal{E}_1\right)$$

where P(s₁) is as follows.

[0131]

[eq. 47]

$$\mathcal{P}_1(s_1) = \left(\Phi(s_1) \rhd [\imath^{s_0}?1].nil\right)/\mathcal{L}(s_1)$$

The above is justified by the following proof by use of the rule of Table 5:

[0132]

[eq. 48]

$$R^\tau_! : \cfrac{R_/ : \cfrac{R^l_\rhd : \cfrac{R_? : \cfrac{1 \in [\![1]\!]}{[\imath^{s_1}?1].\Phi(s_1) \xrightarrow{\imath^{s_1}?1} \Phi(s_1)}}{\Psi(s_1) \xrightarrow{\imath^{s_1}?1} \Phi(s_1) \rhd [\imath^{s_0}?1].nil}}{\mathcal{P}(s_1) \xrightarrow{\imath^{s_1}?1} \mathcal{P}_1(s_1)} \qquad R_! : \cfrac{1 \in [\![1]\!]}{\mathcal{E} \xrightarrow{\imath^{s_1}!1} \mathcal{E}_1}}{\left(\mathcal{P}(s_1) \mid \mathcal{E}\right) \xrightarrow{\tau} \left(\mathcal{P}_1(s_1) \mid \mathcal{E}_1\right)}$$

6. Conclusion

In this paper, first, we have defined the formal process algebra, called Lyee-calculus, that easily and naturally supports the basic concepts of the Lyee methodology. In fact, this calculus can be seen as an abstract machine which is

more suitable to support the Lyee methodology concepts than the Von Newman one. This machine considers a program as a set of molecules that interact together to produce a final result. Secondly, we have shown how this calculus can formalize and simplify both the Lyee vectors together with the whole software generation process. Actually, the routing vectors are no longer needed. Moreover, any pallet function is not unnecessary, and the role of the control function is largely simplified.

We have formalized the whole process of the automatic generation of Lyee software generation. This formalization offers to this methodology a formal semantics that allows one to clearly understand the concepts behind this methodology. Besides, this formal description will be an inevitable start point for many interesting analysis on diverse aspects of this methodology. For instance, to optimize the Lyee generated software or the software generation processes, we need a formal proof that the optimized program is equivalent to its original version. Formal equivalence checking or more generally model checking can be elegantly achieved when using process algebra.

[0133]

As future work, we want to more investigate the semantics of Lyee methodology in order to make it simpler and to generate, from requirements, more reliable and optimized code. For the optimization end, we intend to define a congruence relation between processes of Lyee-calculus and

then using it to prove the correctness of all the eventual optimizations.

[Example 2]

[0134]

<Appendix>

Case Study 2 - Two screens

To show the interaction between many screens, a program constituted of two screens will be described. Fig. 14 is a diagram showing the program screens. When this program is activated, the screen S_1 is displayed. In the screen S_1, the user inputs the word a, and waits until the value of the word b is output. When the button B_2 is pushed, the screen S_2 is displayed, and the user waits until the value of the word b is output. When the button B_0 is pushed in the screen S_2, the program ends.

[0135]

When statements of requirements of the program of Fig. 14 are summarized, the requirement of the screen S_1 is shown in Table 17, and the requirement of the screen S_2 is shown in Table 18.

[0136]

[Table 17]

Word	Definition	Condition	IO	...
a			i	...
b	2*a	a>0	o	...
B_2	S_2	Click	i	...

[0137]

[Table 18]

Word	Definition	Condition	IO	...
e	1+b	b>0	o	...
B_0	S_0	Click	i	...

According to the definition of a generic Lyee program given in the previous section, the Lyee program associated to the requirement given by Tables 17 and 18 is as follows:

[0138]

[eq. 49]

$$SF(s_1) = W_{04}(s_1) \mid W_{03}(s_1) \mid W_{02}(s_1)$$

$$SF(s_2) = W_{04}(s_1) \mid W_{03}(s_1)$$

Notice that there is no W_{02} for the screen s_2 since it does not contain input words other than buttons.

[0139]

[eq. 50]

$$\Phi(s_1) = SF(s_1) \rhd [\imath^{s_1}?1].\Phi(s_1)$$

$$\Phi(s_2) = SF(s_2) \rhd [\imath^{s_2}?1].\Phi(s_2)$$

$$\Psi(s_1, s_2) = ([\imath^{s_1}?1].\Phi(s_1) \mid [\imath^{s_2}?1].\Phi(s_2)) \rhd [\imath^{s_0}?1].nil$$

$$\mathcal{L}(s_1, s_2) = \{d_a, d_b, d_e, d_{B_0}, d_{B_2}\}$$

$$\mathcal{P}(s_1, s_2) = \Psi(s_1, s_2)/\mathcal{L}(s_1, s_2)$$

wherein $W_{02}(s_1)$, $W_{03}(s_1)$, $W_{04}(s_1)$, $W_{03}(s_2)$, and $W_{04}(s_2)$ are

defined as follows. Since the process of R_3 is not essential in the Lyee-calculus in the same manner as in Case Study 1, the process may be omitted.

[0140]

[Table 19]

$W_{02}(s_1) = L_2(a) \mid I_2(a)$	
$L_2(a) \quad = \quad [\imath_2^a?a].$ $[\imath_4^a!a].$ nil	$I_2(a) \quad = \quad [d_a?a].$ $[\imath_2^a!a].$ nil

[0141]

[Table 20]

$W_{03}(s_1) = L_3(b, a > 0) \mid R_3(B_2, click, s_2)$	
$L_3(b, a > 0) \quad = \quad [\imath_4^a?a].$ $[\imath_3^b!(a > 0)].$ nil	$R_3(B_2, click, s_2) \quad = \quad [d_{B_2}?click].$ $[\imath^{s_2}!1].$ nil
$W_{03}(s_2) = L_3(e, b > 0) \mid R_3(B_0, click, s_0)$	
$L_3(e, b > 0) \quad = \quad [\imath_4^b?b].$ $[\imath_3^e!(b > 0)].$ nil	$R_3(B_0, click, s_0) \quad = \quad [d_{B_0}?click].$ $[\imath^{s_0}!1].$ nil

[0142]

[Table 21]

$W_{04}(s_1) = S_4(a) \mid S_4(b) \mid L_4(b, 2*a) \mid O_4(b)$		
$S_4(a) = [j_4^a?a].$ $\qquad S_4^a(y)$ $S_4(b) = [j_4^b?b].$ $\qquad S_4^b(y)$	$L_4(b, 2*a) = [i_3^b!1].$ $[i_4^a?a].$ $[j_4^b!(2*a)].$ nil	$O_4(b) = [i_4^b?b].$ $[d_b!b].$ nil
$W_{04}(s_2) = S_4(e) \mid L_4(e, 1+b) \mid O_4(e)$		
$S_4(e) = [j_4^e?e].$ $\qquad S_4^e(y)$	$L_4(e, 1+b) = [i_3^e!1].$ $[i_4^b?b].$ $[j_4^e!(1+b)].$ nil	$O_4(e) = [i_4^e?e].$ $[d_e!e].$ nil

Suppose now, that the end-user (environment) of this program wants to perform the following sequence of actions to: run the program (activates the screen s_1); give the value 7 to the word a; wait until the value of the word b is obtained; push the button B_2 to go to the screen s_2; wait until the value e is obtained; and then exit the program by pushing the button B_0. This comportment can be captured by the following process ε:

[0143]

[Table 22]

End-user Action	Comments
$\mathcal{E} = [i^{s_1}!1].\mathcal{E}_1$	Activate the program
$\mathcal{E}_1 = [d_a!7].\mathcal{E}_2$	Give the value 7 to the word a
$\mathcal{E}_2 = [d_b?b].\mathcal{E}_3$	Wait for the value of the word b
$\mathcal{E}_3 = [i^{B_2}!click].\mathcal{E}_4$	Push the button B_2 to go to the screen 2
$\mathcal{E}_4 = [d_e?e].\mathcal{E}_5$	Wait for the value of the word e
$\mathcal{E}_5 = [i^{B_0}!click].nil$	Push the button B_0 to exit

Fig. 15 shows the process of the program $P(S_1)$ by the Lyee-calculus and the end-user's process described above.

[0144]

Main steps of the progress of the program are as follows.

The end-user sends a command for activation of the program to thereby activate a control function $\Psi(s_1, s_2)$. The control function $\Psi(s_1, s_2)$ activates a control function $\Phi(s_1)$, and the control function $\Phi(s_1)$ activates the program by running the screen s_1 and $SF(s_1)$.

Synchronization occurs (program activation command is transmitted/received) between $P(s_1, s_2)$ and ε, and the process shifts to a parallel process of $P_1(s_1, s_2)$ and ε_1.

[0145]

[eq. 51]

$$ \mathcal{P}(s_1, s_2) \mid \mathcal{E} \xrightarrow{\tau} \mathcal{P}_1(s_1, s_2) \mid \mathcal{E}_1 $$

where:

[0146]

[eq. 52]

$$ \mathcal{P}_1(s_1, s_2) = \left(\Phi(s_1) \mid [\imath^{s_2}?1].\Phi(s_2) \rhd [\imath^{s_0}?1].nil \right) / \mathcal{L}(s_1, s_2) $$

When the end-user gives (transmits) the value 7 to the word a, $I_2(a)$ in $SF(s_1)$ receives the value 7.

Synchronization occurs (value of the word a is transmitted/received) between $P_1(s_1, s_2)$ and ε_1, and the process shifts to a parallel process of $P_2(s_1, s_2)$ and ε_2.

[0147]

[eq. 53]

$$ \mathcal{P}_1(s_1, s_2) \mid \mathcal{E}_1 \xrightarrow{\tau} \mathcal{P}_2(s_1, s_2) \mid \mathcal{E}_2 $$

where:

[0148]

[eq. 54]

$$\mathcal{P}_2(s_1, s_2) \;=\; ((((W_{04}(s_1) \mid [\imath_2^a!7].nil \mid L_2(a) \mid W_{03}(s_1)) \rhd [\imath^{s_1}?1].\Phi(s_1) \mid [\imath^{s_2}?1].\Phi(s_2)) \\ \rhd[\imath^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

The value 7 of a is stored in the memory $S_4(a)$: This will be done in two steps. First, the value 7 of a is communicated to $L_2(a)$ and second $L_2(a)$ will communicate this value to the memory $S_4(a)$.

The first step is as follows: synchronization occurs in $P_2(s_1, s_2)$ (the value 7 of the word a is transmitted/received between $I_2(a)$ and $L_2(a)$), and the process shifts to a parallel process of $P_3(s_1, s_2)$ and ε_2.

[0149]

[eq. 55]

$$\mathcal{P}_2(s_1, s_2) \mid \mathcal{E}_2 \;\xrightarrow{\tau}\; \mathcal{P}_3(s_1, s_2) \mid \mathcal{E}_2$$

where:

[0150]

[eq. 56]

$$\mathcal{P}_3(s_1, s_2) \;=\; (((W_{04}(s_1) \mid [\jmath_4^a!7].nil \mid W_{03}(s_1)) \rhd [\imath^{s_1}?1].\Phi(s_1) \mid [\imath^{s_2}?1].\Phi(s_2)) \\ \rhd[\imath^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

In the second, the value 7 of a will be stored in the memory $S_4(a)$ of W_{04} (the initial value is stored in $S_4(a)$ to thereby initialize the memory). Synchronization occurs in $P_3(S_1, S_2)$ (value 7 of the word a is transmitted/received between $L_2(a)$ and $S_4(a)$), and the process shifts to a

parallel process of P₄(s₁, s₂) and ℰ₂.

[0151]

[eq. 57]

$$\mathcal{P}_3(s_1, s_2) \mid \mathcal{E}_2 \xrightarrow{\tau} \mathcal{P}_4(s_1, s_2) \mid \mathcal{E}_2$$

where:

[0152]

[eq. 58]

$$\mathcal{P}_4(s_1, s_2) = (((S_4^a(7) \mid S_4(b) \mid L_4(b, 2*a) \mid O_4(b) \mid W_{03}(s_1)) \rhd [\imath^{s_1}?1].\Phi(s_1) \mid [\imath^{s_2}?1].\Phi(s_2))$$
$$\rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

The value 7 of a is communicated from the initialized
memory S₄ of a to L₃(b, a>0) to compute the calculation
conditions of b. Synchronization occurs in P₄(s₁, s₂)
(transmission/reception between S₄(a) and L₃(b, a>0)), and
the process shifts to a parallel process of P₅(s₁, s₂) and ℰ₂.

[0153]

[eq. 59]

$$\mathcal{P}_4(s_1, s_2) \mid \mathcal{E}_2 \xrightarrow{\tau} \mathcal{P}_5(s_1, s_2) \mid \mathcal{E}_2$$

where:

[0154]

[eq. 60]

$$\mathcal{P}_5(s_1, s_2) = (((S_4^a(7) \mid S_4(b) \mid L_4(b, 2*a) \mid O_4(b) \mid [\imath_3^b!(7 > 0)].nil \mid R_3(B_2, s_2, click))$$
$$\rhd [\imath^{s_1}?1].\Phi(s_1) \mid [\imath^{s_2}?1].\Phi(s_2)) \rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

The calculation conditions of b is evaluated (a>0 is
evaluated to be true (1)), and the value 1 is communicated to

- 69 -

$L_4(b, 2*a)$. Synchronization occurs in $P_5(s_1, s_2)$ (transmission/reception of a true/false value 1 between $L_3(b, a>0)$ and $L_4(b, 2*a)$), and the process shifts to a parallel process of $P_6(s_1, s_2)$ and $\mathcal{E}_2$.

[0155]

[eq. 61]

$$\mathcal{P}_5(s_1, s_2) \mid \mathcal{E}_2 \xrightarrow{\tau} \mathcal{P}_6(s_1, s_2) \mid \mathcal{E}_2$$

where:

[0156]

[eq. 62]

$$\mathcal{P}_6(s_1, s_2) = (((S_4^a(7) \mid S_4(b) \mid [j_4^a!(2*a)].nil \mid O_4(b) \mid R_3(B_2, s_2, click)) \rhd [\imath^{s_1}?1].\Phi(s_1) \mid [\imath^{s_2}?1].\Phi(s_2)) \rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

The value 7 of a is communicated from the memory S_4 of a to $L_4(b, 2*a)$, a definition "2*a" of b is computed, and a value 14 of b is stored in a memory $S_4(b)$ ($S_4(b)$ is initialized). Synchronization occurs in $P_6(s_1, s_2)$ (transmission/reception between the memory S_4 of a and $L_4(b, 2*a)$ and between $L_4(b, 2*a)$ and the memory S_4 of b), and the process shifts to a parallel process of $P_7(s_1, s_2)$ and $\mathcal{E}_2$.

[0157]

[eq. 63]

$$\mathcal{P}_6(s_1, s_2) \mid \mathcal{E}_2 \xrightarrow{\tau} \mathcal{P}_7(s_1, s_2) \mid \mathcal{E}_2$$

where:

[0158]

[eq. 64]

$$\mathcal{P}_7(s_1, s_2) = (((S_4^a(7) \mid S_4^b(14) \mid O_4(b)) \triangleright [\imath^{s_1}?1].nil \mid R_3(B_2, s_2, click))$$
$$\triangleright [\imath^{s_1}?1].\Phi(s_1) \mid [\imath^{s_2}?1].\Phi(s_2)) \triangleright [\imath^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

The value 14 of b is communicated from the memory S_4 of b to an output process O_4(b). The b value is communicated to the environment by O_4(b), and the end user receives the value of b. Synchronization occurs in P_7(s_1, s_2) (transmission/reception between L_4(b, 2*a) and O_4(b)), synchronization occurs between P_7(s_1, s_2) and ε_2 (transmission/reception of value of word b), and the process shifts to P_8(s_1, s_2) and ε_3.

At this time, processes belonging to SF(s_1) being activated are S_4 which holds values of a and b, and R3(B_0, click, s_0).

[0159]

[eq. 65]

$$\mathcal{P}_7(s_1, s_2) \mid \mathcal{E}_2 \xrightarrow{\tau} \mathcal{P}_8(s_1, s_2) \mid \mathcal{E}_3$$

where:

[0160]

[eq. 66]

$$\mathcal{P}_8(s_1, s_2) = (((S_4^a(7) \mid S_4^b(14) \mid R_3(B_2, s_2, click)) \triangleright [\imath^{s_1}?1].\Phi(s_1) \mid [\imath^{s_2}?1].\Phi(s_2))$$
$$\triangleright [\imath^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

The end-user pushes the button B_2 in the screen s_1 to go to screen s_2. R3(B_2, click, s_2) receives the button-B_2-clicked value transmitted from ε_3, and R3(B_2, click, s_2) evaluates its own calculation conditions to transmit a true value of 1 to a control function Ψ(s_1, s_2). The control function Ψ(s_1, s_2) which has received the value 1 activates

the control function $\Phi(s_2)$. The control function $\Phi(s_2)$ which has received the value 1 activates $SF(s_2)$ and the screen s_2. There occur: synchronization between $P_8(s_1, s_2)$ and $\mathcal{E}_3$ (transmission/reception of the value clicked by the button B_2); and synchronization in $P_8(s_1, s_2)$ (transmission/reception between $R3(B_2, click, s_2)$ and $\Psi(s_1, s_2)$ and the control function $\Phi(s_2)$), and the process shifts to a parallel process of $P_9(s_1, s_2)$ and $\mathcal{E}_4$.

[0161]

[eq. 67]

$$\mathcal{P}_8(s_1, s_2) \mid \mathcal{E}_3 \xrightarrow{\tau} \mathcal{P}_9(s_1, s_2) \mid \mathcal{E}_4$$

where:

[0162]

[eq. 68]

$$\mathcal{P}_9(s_1, s_2) = (((S_4^a(7) \mid S_4^b(14)) \rhd [\imath^{s_1}?1].\Phi(s_1) \mid \Phi(s_2)) \rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

To compute the condition of e, the memory S_4 of b transmits value 14 to $L_3(e, b>0)$. Synchronization occurs in $P_9(s_1, s_2)$ (transmission/reception between the memory S_4 of a and $L_3(e, b>0)$), and the process shifts to a parallel process of $P_{10}(s_1, s_2)$ and $\mathcal{E}_4$.

[0163]

[eq. 69]

$$\mathcal{P}_9(s_1, s_2) \mid \mathcal{E}_4 \xrightarrow{\tau} \mathcal{P}_{10}(s_1, s_2) \mid \mathcal{E}_4$$

where:

[0164]

[eq. 70]

$$\mathcal{P}_{10}(s_1, s_2) = (((S_4^a(7) \mid S_4^b(14)) \triangleright [\imath^{s_1}?1].\Phi(s_1) \mid (W_{04}(s_2) \mid [\imath_3^e(14 > 0)].nil \mid \\ R_3(B_0, s_0, click)) \triangleright [\imath^{s_1}?1].\Phi(s_1)) \triangleright [\imath^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

The calculation condition b>0 of e is evaluated to be true (1), a true value 1 is transmitted from L_3(e, b>0) to L_4(e, 1+b), and the value of e is computed. Synchronization occurs in P_{10}(s_1, s_2) (transmission/reception between L_3(e, b>0) and L_4(e, 1+b), and the process shifts to a parallel process of P_{11}(s_1, s_2) and ε_4.

[0165]

[eq. 71]

$$\mathcal{P}_{10}(s_1, s_2) \mid \mathcal{E}_4 \xrightarrow{\tau} \mathcal{P}_{11}(s_1, s_2) \mid \mathcal{E}_4$$

where:

[0166]

[eq. 72]

$$\mathcal{P}_{11}(s_1, s_2) = (((S_4^a(7) \mid S_4^b(14)) \triangleright [\imath^{s_1}?1].\Phi(s_1) \mid (S_4(e) \mid [\jmath_4^e(1 + 14)].nil \mid O_4(e) \mid \\ R_3(B_0, s_0, click)) \triangleright [\imath^{s_1}?1].\Phi(s_1)) \triangleright [\imath^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

Store the value of e from L_4(e, 1+b) in the memory S_4(b) of e (S_4(b) is initialized). Synchronization in P_{11}(s_1, s_2) (transmission/reception between L_4(e) and the memory S_4 of e), and the process shifts to a parallel process of P_{12}(s_1, s_2) and ε_4.

[0167]

[eq. 73]

$$\mathcal{P}_{11}(s_1, s_2) \mid \mathcal{E}_4 \xrightarrow{\tau} \mathcal{P}_{12}(s_1, s_2) \mid \mathcal{E}_4$$

where:

[0168]

[eq. 74]

$$\mathcal{P}_{12}(s_1, s_2) = (((S_4^a(7) \mid S_4^b(14)) \triangleright [\imath^{s_1}?1].\Phi(s_1) \mid (S_4^e(15) \mid O_4(e) \mid$$
$$R_3(B_0, s_0, click)) \triangleright [\imath^{s_1}?1].\Phi(s_1)) \triangleright [\imath^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

Transmit a value 15 of e from the memory S_4(b) of e to O_4(e), and output the value of e from O_4(e) to the end-user $\mathcal{E}_4$. Synchronization occurs between P_{11}(s_1, s_2) and $\mathcal{E}_4$ (transmission/reception of the value 14 of the word e), and the process shifts to a parallel process of P_{12}(s_1, s_2) and $\mathcal{E}_5$.

[0169]

[eq. 75]

$$\mathcal{P}_{11}(s_1, s_2) \mid \mathcal{E}_4 \xrightarrow{\tau} \mathcal{P}_{12}(s_1, s_2) \mid \mathcal{E}_5$$

where:

[0170]

[eq. 76]

$$\mathcal{P}_{12}(s_1, s_2) = (((S_4^a(7) \mid S_4^b(14)) \triangleright [\imath^{s_1}?1].\Phi(s_1) \mid (S_4^e(15) \mid$$
$$R_3(B_0, s_0, click)) \triangleright [\imath^{s_1}?1].\Phi(s_1)) \triangleright [\imath^{s_0}?1].nil)/\mathcal{L}(s_1, s_2)$$

The end-user pushes B_0 to exit the program, and R3(B_0, click, s_0) receives the value click of the button B_0 from the end-user $\mathcal{E}_5$, and transmit the value to a control function ψ(s_1, s_2). The control function ψ(s_1, s_2) ends Φ(s_1) and Φ(s_2) before the function itself ends, and Φ(s_1) ends SF(s_1)

and $\Phi(s_2)$ ends $SF(s_2)$, respectively, before they end. Synchronization occurs between $P_{12}(s_1, s_2)$ and $\mathcal{E}_5$ (transmission/reception of Click value), next synchronization occurs in $P_{12}(s_1, s_2)$ (transmission/reception of R3(B_0, click, s_0), $\Psi(s_1, s_2)$, and the control functions $\Phi(s_1)$ and $\Phi(s_2)$), and all the processes shift to nil.

[0171]

[eq. 77]

$$\mathcal{P}_{12}(s_1,s_2) \mid \mathcal{E}_5 \quad \xrightarrow{\tau} \quad (((S_4^a(7) \mid S_4^b(14)) \rhd [\imath^{s_1}?1].\Phi(s_1) \mid (S_4^c(15) \mid [\imath^{s_0}?1].nil) \rhd [\imath^{s_1}?1].\Phi(s_1)) \rhd [\imath^{s_0}?1].nil)/\mathcal{L}(s_1,s_2)$$
$$\xrightarrow{\tau} \quad nil$$

It is to be noted that the present invention is not limited to the above-described embodiments and examples, and various modifications are possible in a region of technical idea of the present invention. The present invention can be realized as, for example, a business method, a software development device, a software development support device, a software development and management device, software for realizing these functions in a computer, a recording medium or an special purpose machine on which the software is installed or the like. Furthermore, needless to say, as described above, the present invention can be realized as a method, software having such function, a device or a tool (including the software itself) on which the software is installed, or a system.

[0172]

For example, Fig. 16 is a functional block diagram showing, as one of different embodiments of the present

invention, a constitution including a function in a case
where the present invention is embodied as one of a program
(software) for producing "software of a development object",
a program generation device, a program processing device, a
tool (including both of tools as a device and software), a
software development device, a software development support
device, and a software development and management device. As
shown in the drawing, the present invention is one of the
program (software), the program generating device, the
program processing device, the tool (including both of the
tools as the device and the software), the software
development device, the software development support device,
and the software development and management device. The
present invention includes an overall control section 1601,
a statement information assignment section 1602, a control
function Φ placement section 1603, a control function Ψ
placement section 1604, and an information memory section
1605.

[0173]

The overall control section 1601 has a function of
performing the whole operation control, timing control,
input/output control and the like of the present program
(software), the program generating device, the program
processing device, the tool (including both of the tools as
the device and the software), the software development device,
the software development support device, and the software
development and management device. The overall control

section is realized as an special purpose chip having the function, an special purpose circuit, software (also including the software as the tool) for allowing the computer to perform the function, a recording medium in which the software is recorded, or a processing device, a management device, or a tool on which the recording medium is mounted.

[0174]

The statement information assignment section 1602 has a function of assigning information into undefined parts of models of signification vectors (L_2, L_3, L_4) and action vectors (I_2, O_4, S_4), each of which is a module of a process cell having input/output channels based on the Lyee-calculus, and such assigned information is formed from a user requirement to be implemented as one program into statements, and each of such statements declared by word and grouped by logical record includes a word name, a definition expression, a condition for executing the definition expression, an input/output attribute, and an attribute of a value of the word. The section is realized as a special purpose chip or a special purpose circuit having the function, software (including software as a tool) for allowing a computer to perform the function, a recording medium in which the software is recorded, or a processing device, a management device or a tool on which the recording medium is mounted.

[0175]

The control function Φ placement section 1603 has a function of associating control function modules Φ so that a

single control function module Φ controls a single set of the signification vectors and the action vectors which are grouped on the set condition that interactions among it are caused by a command from the same screen. The section is realized as a special purpose chip or a special purpose circuit having the function, software (including software as a tool) for allowing a computer to perform the function, a recording medium in which the software is recorded, or a processing device, a management device or a tool on which the recording medium is mounted.

[0176]

The control function Ψ placement section 1604 has a function of associating a control function module Ψ so that a sigle control function Ψ controls the control function modules Φ. The section is realized as a special purpose chip or a special purpose circuit having the function,, software (including software as a tool) for allowing a computer to perform the function, a recording medium in which the software is recorded, or a processing device, a management device or a tool on which the recording medium is mounted.

[0177]

The information memory section 1605 is used as a storage of: program information; further various control information; the models of the signification vectors (L2, L3, L4) and the action vectors (I2, O4, S4), each of which is formed as a module of a process cell; the model of the control function module Φ; the model of the control function module Ψ; a

target program and the like. Moreover, the section is also used as a memory for temporary storage.

[0178]

Fig. 17 is a flowchart showing an operation of the present invention implemented as one the followings which has the above-described constitution: program (software) a program generating device, a program processing device, a tool (including both of tools as the device and the software), a software development device, a software development support device, and a software development and management device.

[0179]

As shown in the drawing, first the statement information assignment section 1602 inserts the information into the undefined parts of the models of the signification vectors (L2, L3, L4) and the action vectors (I2, O4, S4), each of which is formed as the module of the process cell having the input/output channels based on the Lyee-calculus; such information is formed from the user requirement to be implemented as one program into the statements; each of such statements declared by word and grouped by logical record includes a word name, a definition expression, a condition for executing the definition expression, an input/output attribute, and an attribute of a value of the word(step 1701).

[0180]

Then, the control function Φ placement section 1603 associates control function modules Φ so that a single control function module Φ controls a single set of the

signification vectors and the action vectors which are grouped on the set conditions that interactions among it are caused by the command from the same screen (step 1702).

[0181]

Next, the control function Ψ placement section 1604 associates one control function module Ψ so that a single control function Ψ controls the control function modules Φ (step 1703).

[0182]

Therefore, according to the present invention constituted as described above, the requirement is defined on the basis of the theoretical system unique to the present invention, and the defined requirement is assigned into the undefined parts of the models of the signification vectors (L2, L3, L4) and the action vectors (I2, O4, S4), each of which is the module of the process cell having the structure unique to the present invention. Therefore, desired software can be obtained without humane/personal elements.

[0183]

A program (software) for producing "software of a development object", a program generation device, a program processing device, a tool (including the tools as both the device and the software), a software development device, a software development support device, or a software development and management device as one of different embodying units of the present invention may include: statement information assignment means for assigning

information into undefined parts of the models of the signification vectors (L_2, L_3, L_4) and the action vectors (I_2, O_4, S_4), each of which is formed into the module of a process cell having input/output channels based on the Lyee-calculus, where the information is formed from a user requirement to be mounted as one program into statements, and each of such statements declared by word and grouped by logical record includes a word name, a definition expression, a condition for executing the definition expression, an input/output attribute, and an attribute of a value of the word; control function Φ placement means for associating control function modules Φ so that a single control function module Φ controls a single set of the signification vectors and the action vectors grouped on the set condition that interactions among it are caused by a command from the same screen; and control function Ψ placement means for associating one control function module Ψ so as to control the control function modules Φ.

[0184]

The present invention is also realized by the software produced by the above-described "method of producing software of the development object", a recording medium on which the software is mounted, or a device (hardware) on which the software is mounted. In this case, the present invention may be constituted of: modules made from the models of the signification vectors (L_2, L_3, L_4) and the action vectors (I_2, O_4, S_4), each of which is the module of a process cell having

input/output channels based the Lyee-calculus, and such modules are made by assigning information to undefined parts of such models, and such information is formed from a user requirement to be implemented as one program into statements, and each of such statements declared by word and grouped by logical record has a word name, a definition expression, a condition for executing the definition expression, an input/output attribute, and an attribute of a value of the word; control function modules Φ associated so that a single control function module Φ controls a set of the above mentioned modules grouped based on the set condition that interactions among it are caused by a command from the same screen; and a control function module Ψ associated so that a single control function module Ψ controls the control funcion modules Φ.

[0185]

Furthermore, the present invention is also realized as software which is a model of a software code for use in producing software by the above-described "method of producing the software of the development object", a recording medium on which the software is mounted, or a device (hardware) on which the software is mounted. In this case, the present invention may be constituted of: models of the signification vectors (L_2, L_3, L_4) and the action vectors (I_2, O_4, S_4), each of which is the module of a process cell having input/output channels based the Lyee-calculus, and such modules have undefined parts where information should be

assigned, and such information is formed from a user
requirement to be implemented as one program into statements,
where each of such statements declared by word and grouped by
logical record has a word name, a definition expression, a
condition for executing the definition expression, an
input/output attribute, and an attribute of a value of the; a
model of control function module Φ which has a function that
a single control function module Φ controls a set of the
signification vectors and the action vectors which have the
statement information in the unidentified parts, and such set
is grouped based on the set condition that interactions among
it are caused by a command from the same screen; and a model
of control function module Ψ which has a function that a
single control function module Ψ controls the control
function modeuls Φ.

[0186]

Furthermore, the present invention can be realized by
the above-described "method of producing the software of the
development object" as any of a method of extracting
information (document (paper, data)) extracted from a
requirement, the information (document (paper, data))
extracted by such extraction method, a method of using the
extracted information, an information recording medium on
which these information is mounted, software in which an
information extracting/using method is coded, and a recording
medium/device (hardware) on which the software is mounted.
In this case, the present invention may be realized as

information extracted from a software development requirement, including: information formed from a user requirement to be mounted as one program into statements, each of which is declared by word and grouped by logical record and has a word name, a definition expression, a condition for executing the definition expression, an input/output attribute, and an attribute of a value of the word, and such information being to be assigned into undefined parts of the models of the signification vectors (L_2, L_3, L_4) and the action vectors (I_2, O_4, S_4), each of which is the module of a process cell having input/output channels based on the Lyee-calculus; information for associating control function modules Φ so that a single control function module Φ controls a single set of the signification vectors and the action vectors, which have the above mentioned information in their unidentified parts, and such set is grouped on the set condition that interactions among it are caused by a command from the same screen; and information for associating a control function module Ψ so as to control the control function modules Φ.

[0187]

Furthermore, the present invention permits various modifications, additions, replacements, enlargements, reductions and the like in identical and equivalent regions of the technical idea. The value of the present invention is not lowered at all even in a case where the device, the method, the software, or the system produced using the present invention is mounted on a secondary product, and

commercialized.

[Industrial Applicability]

[0188]

A wide range of software problems related to different fields are efficiently dealt with, and development time, maintenance time and documentation volume can be considerably reduced by using Lyee (as much as 70 to 80%) as compared with conventional methodologies.

[Brief Description of the Drawings]

[0189]

Fig. 1 is a schematic diagram showing a concept of Cell in one embodiment of the present invention;

Fig. 2 is a diagram showing Example of Interacting Processes in one embodiment of the present invention;

Fig. 3 is a schematic diagram showing a concept of requirement execution in one embodiment of the present invention;

Fig. 4 is a schematic diagram showing a concept of Lyee Pallets in one embodiment of the present invention;

Fig. 5 is a schematic diagram showing a concept of Basic Structure (Scenario Function) in one embodiment of the present invention;

Fig. 6 is a schematic diagram showing a concept of Predicate Vector in one embodiment of the present invention;

Fig. 7 is a schematic diagram showing concepts of the Predicate Vectors of L4_a and L4_b in one embodiment of the present invention;

Fig. 8 is a schematic diagram showing concepts of the Predicate Vectors of L3_a and L3_b in one embodiment of the present invention;

Fig. 9 is a schematic diagram showing a concept of Screen Interaction in one embodiment of the present invention;

Fig. 10 is a schematic diagram showing Processes Route Diagram in one embodiment of the present invention;

Fig. 11 is a schematic diagram showing Case Study - One Screen in one embodiment of the present invention (Embodiment 1);

Fig. 12 is an explanatory view showing a process of program $P(S_1)$ and end-user's process by the Lyee-calculus in one embodiment of the present invention;

Fig. 13 is a schematic diagram showing Interaction between processes in one embodiment of the present invention;

Fig. 14 is a schematic diagram showing Case Study - Two Screens in one embodiment of the present invention (Embodiment 2);

Fig. 15 is an explanatory view showing a process of program $P(S_1)$ and end-user's process by the Lyee-calculus in one embodiment of the present invention;

Fig. 16 is a functional block diagram showing a constitution of functional units in one of different embodiments of the present invention in a case where the present invention is embodied as one of a program (software) for producing "software of a development object", a program

generation device, a program processing device, a tool (including both of tools as the device and the software), a software development device, a software development support device, and a software development and management device; and

Fig. 17 is a flowchart showing an operation of the present invention embodied as one of the program (software), the program generation device, the program processing device, the tool (including both of tools as the device and the software), the software development device, the software development support device, and the software development and management device constituted as described above in one embodiment of the present invention.

[Description of Symbols]

[0190]

W04   W04 pallet

W02   W02 pallet

W03   W03 pallet

1601   overall control section

1602   statement information assignment section

1603   control function Φ placement section

1604   control function Ψ placement section

1605   information memory section